DYNAMIC BUFFER MANAGEMENT POLICY FOR SHARED MEMORY PACKET SWITCHES BY EMPLOYING PER-QUEUE THRESHOLDS

BORAN GAZI

PhD

2006

DYNAMIC BUFFER MANAGEMENT POLICY FOR SHARED MEMORY PACKET SWITCHES BY EMPLOYING PER-QUEUE THRESHOLDS

BORAN GAZI

A thesis submitted in partial fulfilment of the requirements of the University of Northumbria at Newcastle for the degree of Doctor of Philosophy

Research undertaken in the School of Computing, Engineering and Information Sciences

November 2006

ABSTRACT

One of the main problems concerning high-performance communications networks is the unavoidable congestion in network nodes. Network traffic is normally characterised as "bursty", which may use up network resources during peak periods. As a consequence end-user applications are subject to end-to-end delays and disruptions. Simultaneous transmission of packets on a finite bandwidth channel might result in contentions, where one or more packets are refrained from entering the transmission channel resulting in packet losses. Hence, the motivations of this thesis are two-fold: investigation and evaluation of switch architectures with electronic and optical buffers, and the development and evaluation of an improved dynamic threshold policy for shared memory switch architecture. In this work, switch architectures based on modular designs are evaluated, with simulation results showing that modular switch structures, i.e. multistage interconnection networks with optical delay line buffers, offer packet loss rate, throughput and average delay time similar to their electronic counterparts. Such optical architectures emulate prime features of shared memory switch architecture under general traffic conditions. Although the shared memory switch architecture is superior to other buffering approaches, but its performance is inadequate under imbalanced input traffic. Here its limiting features are investigated by means of numerical analysis. Different buffer management schemes, namely static thresholds, dynamic thresholds, pre-emptive, adaptive control, are investigated by using the Markov simulation model. An improved dynamic buffer management policy, decay function threshold (DFT) policy, is proposed and it is compared with the dynamic thresholds (DT), partial sharing partial partitioning

(PSPP) and dynamic queue thresholds (DQT) buffer management policies by using bursty traffic source models, such as interrupted Poisson process (IPP), by means of simulations. Simulation results show that proposed policy is as good as well-known dynamic thresholds policy in the presence of best-effort traffic and offers improved packet loss performance when multicast traffic is considered. An integration framework for dynamic buffer management and bandwidth scheduling is also presented in this study. This framework employs loosely coupled buffer management and scheduling (weighted round robin, weighted fair queueing etc.) providing support for quality of service traffic. Conducted tests show that this framework matches the best-effort packet loss performance of dynamic thresholds policy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

c	DFT share parameter
$\overline{d}$	Average waiting time
$B(t)$	Size of unused buffer space at time t
H	Number of heavily loaded ports
H_p	Hotspot probability
L_p	Load on heavily loaded port
λ	Arrival rate
M	Buffer capacity (the number of fixed length packets)
N	Switch dimension
p	Traffic load
p_0	Output utilisation
π_{ij}	State probability
$q_i(t)$	Queue length of output port i at time t
$Q(t)$	Buffer occupancy at time t
$T(t)$	Queue threshold at time t
$T_i(t)$	Per-queue threshold
$T_{ij}(t)$	Priority queue threshold
T_{ON}	Duration of ON state
T_{OFF}	Duration of OFF state
$\overline{W}$	Steady-state delay
$X(t_n)$	Observation at time t
δ	Average delay

θ	Average buffer utilisation
μ	Departure rate

ACKNOWLEDGEMENTS

I would like to thank those people who directly and indirectly contributed to this study. This work would not have been possible without the support and encouragement of people around me: Prof. Fary Z. Ghassemlooy, who provided guidance and whom I involved in endless discussions from the first day I joined his research group until the submission of this thesis. His determination and contributions to research inspired me. My parents and brother, who provided endless support to get through financial difficulties, for helping me have a higher education. Last but not least, my partner Jingjing, who stood by my side during difficult times, for providing love and encouragement. Finally the rest, who kept asking me whether I finished my studies; such questions have been my main motivation.

DECLARATION

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my own work.

Name: BORAN GAZI

Signature:

Date: 17/04/2007

CHAPTER 1

INTRODUCTION

1.1. Motivation

Communications networks are continuously experiencing an upsurge in the demand for bandwidth due to an increase in the number of end-users and an increase in the bandwidth capacity requirements of end-user applications. Edholm's law of bandwidth suggests that data rates are doubling every eighteen months (Cherry 2004). Requirement for bandwidth is mainly driven by end-user applications, such as teleconferencing, high definition television (HDTV), distance learning etc., increase in the number of online users and mobility. In recent years, developments in data communication systems, such as photonics and wireless communications, have provided the opportunity to enhance the network capacity and cope with end-user demands.

Whilst emerging toward ultra-fast networks, congestion in network nodes still remains as a crucial bottleneck. In a packet-switched network, each packet has to traverse a number of nodes to reach its destination. Every network node carries out mainly three functions: header processing, buffering and switching/routing. Header processing is required to extract packet header information. Meanwhile switching/routing ensures that packets leave from the correct output ports based on the information extracted from the packet headers. Due to unscheduled packet arrivals, two or more packets arriving simultaneously at a network node may be destined for the same output port. In such situation, only one packet is allowed to

leave from the correct output port whilst the rest of the packets are dropped resulting in packet losses. This is known as contention. Various techniques have been proposed in order to avoid this problem: (i) deflection routing, (ii) wavelength conversion and (iii) buffering. Deflection routing ensures that one of the contending packets leave from the correct output port and the rest of the packets are sent to available output ports. Wavelength conversion technique allows all packets to leave the switch from the correct output port in different wavelengths. In buffering, only one of the packets is allowed to leave the switch and contending packets wait in switch buffers until contentions are resolved. Buffering can be considered as virtual bandwidth used to accommodate packets when the output port channel is not fast enough to allow more than one packet to be transmitted simultaneously.

Due to bursty nature of network traffic, switch buffers often become congested temporarily and inadequate amount of buffer space is left to accommodate incoming bursts. Buffer depths can be easily increased to resolve this problem. Nevertheless, delay sensitive applications, such as teleconferencing, IP-telephony, cannot cope with such delays. If re-transmission is feasible, packets can be dropped to ease congestion and transmissions can be reinitiated for those packets dropped from the end of the first-in-first-out (FIFO) queues. Therefore, a crucial question is raised: is it feasible to drop packets for re-transmission or store temporarily until contentions are resolved?

Switch architectures can be classified according to the positions of FIFO buffers (Guizani and Al-Fuqaha 2001). These are input queuing, output queuing and shared buffering. Input queuing exploits the head of line blocking problem and for very large switch sizes ($N \rightarrow \infty$), switch throughput is limited to ($2 - \sqrt{2}$) (Karol et al. 1987).

Switches with output queues have their fixed-length FIFO queues designated at the switch output ports. Output queuing achieves optimal throughput-delay performance under uniform input traffic (Engbersen 2003). Shared buffer switch architectures consist of a common buffer space entirely shared among N output ports in an $N \times N$ non-blocking packet switch (Engbersen 2003). Shared buffering achieves similar performances as output queueing under uniform input traffic. The main benefit of shared buffering is its ability to accommodate buffer space in a flexible manner. Besides, buffer space required to achieve similar packet loss performance is smaller than that of output queueing.

Shared buffer switches, also known as shared memory switch, are widely used in high performance packet switching network nodes. Nevertheless, packet losses are extreme when the input traffic is unevenly distributed among output ports (Wei et al. 1991). Even though the input traffic is symmetric, loads fluctuate temporarily for each output port. As a consequence, active queues dominate the usage of buffer space and no buffering becomes available for lightly loaded queues. Therefore, buffer management (buffer allocation) policy is essential to maintain fairness among output ports. Buffer allocation policies can be categorised into two main groups (Arpaci and Copeland 2000): pre-emptive and non-preemptive. Pre-emptive policies, such as drop on demand (Wei et al. 1991) and push-out with thresholds (Cidon et al. 1995), replace those packets that are already in the switch buffers. Pre-emptive policies are naturally adaptive and far better than non-preemptive policies as packets are stored in the switch buffers until when the loss is unavoidable. However, replacing a packet, which is already in the buffer memory, is not feasible and also searching for the longest queue has the complexity of $O(N)$. Non-preemptive policies prevent packets from

entering the switch buffers. These are further classified as static thresholds, dynamic thresholds and optimising control. Static thresholds, such as sharing with a minimum allocation (SMA), sharing with maximum queue lengths (SMXQ) and sharing with a maximum queue and minimum allocation (SMQMA), are optimised for those conditions where traffic phase do not change significantly (Arpaci and Copeland 2000). For this reason they perform poorly in dynamic environments. Dynamic policies, such as dynamic thresholds (DT) (Choudhury and Hahne 1998), dynamic queue thresholds (DQT) (Fan et al. 1999), harmonic buffer management (Kesselman and Mansour 2002), and partial sharing partial partitioning (PSPP) (Chu et al. 2002), have high awareness of network dynamics and adopt thresholds dynamically. Optimising control schemes repeatedly try to optimise the switch performance (e.g. maximise throughput, minimise packet loss etc.) by employing online optimisation techniques (Ascia et al. 2002).

1.2. Scope and Goal of the Research

The study investigates electronic and all-optical packet switch architectures that employ packet buffers to overcome packet losses due to contentions that occur when two or more packets are destined for the same output port and scheduled to depart in the same instance (time-slot). This study is limited with coarse performance measures, such as packet loss rate (packet loss rate is the ratio of the number of packets lost to the number of packets injected to the system), throughput (a measure of the amount of data transferred in a specific amount of time) and average delay (average elapsed time between the instance when the data is sent and when it is received at the other end), rather than physical features, such as signal profile, signal-noise ratio etc. In the

literature, analytical models were used in modelling the packet switch performance. Assumptions made and traffic models used are inadequate in effectively modelling the network. For this reason, simulation methods are used in this study to observe the main features of buffering schemes and buffer management policies. Traffic models are also considered in this study, as traffic pattern and distribution have great impact on network performance.

The aim of this study is to develop and evaluate dynamic buffer management policy for shared buffer memory packet switch architecture. To achieve this, the main goals of this study are set as follows:

- Investigate electronic and all-optical packet switch architectures that employ buffering to overcome the contention problem.
- Compare and evaluate performances of various buffering schemes used as node architectures in scalable packet switch architecture (multi-stage interconnection network).
- Investigate and analyse the common features of buffer management policies used in shared memory packet switch architectures. Investigate the effect of input traffic in this type of switch architecture.
- Develop an improved dynamic buffer management policy for the shared memory packet switch architecture.
- Simulate and evaluate the proposed buffer management policy in contrast to existing dynamic policies in the presence of best-effort, multicast and priority traffic models.

1.3. Organisation of the Thesis

The significance of employing buffer memories in packet switched networks and its impact on high-performance packet switching is addressed in this chapter. The rest of the thesis consists of four main parts: (i) literature survey, (ii) description and performance evaluation of the scalable switch architecture that employs off-the-shelf electronic and all-optical binary switch architectures, (iii) investigation and analysis of buffer allocation policies in shared memory packet switch architectures, (iv) description of the proposed dynamic buffer allocation policy and its performance evaluation in contrast to existing policies via simulations.

Chapter two provides a review of literature relevant to this research. This chapter presents an in-depth investigation of proposed electronic and all-optical packet switch architectures that employ packet buffers. Electronic packet switches make use of random access memories to store packets, whereas their all-optical counterparts use optical delay lines to delay packets temporarily. The positions of first-in-first-out (FIFO) queues in electronic packet switches relates to the switch performance. In optical domain, despite the fact that optical signals being faster, switch design patterns concede limitations of delay lines, e.g. the number of recirculations in an optical delay line yields in degradation of the optical signal.

Chapter three is dedicated to developing scalable switch architecture with distributed buffers and evaluating its performance by means of simulations. Large scale switches can be built from binary switches by following an interconnection pattern such as butterfly permutation, known as multi-stage interconnection network (MIN). Due to

architectural limitations, e.g. single path between each input-output pair, MINs are prone to high packet losses. In order to overcome this problem some of the buffering strategies discussed in Chapter two, output queueing, shared buffering, virtual output queueing (VOQ) and 2×2 cascaded optical delay architectures, will be considered to achieve scalable switch architecture with distributed buffers. Performance characteristics of different buffering strategies used with MIN, i.e. scalability and suitability, will then be analysed through simulations. An all-optical implementation of MIN with optical buffers is analysed for the first time in this chapter.

Chapter four studies the existing buffer allocation policies studied in the literature to overcome the degraded performance issues due to bursty and imbalanced input traffic. Shared memory packet switch architecture is described, and two extreme allocation policies, namely complete sharing (CS) and complete partitioning (CP), are then analysed in the presence of imbalanced input traffic through numerical analysis. Buffer allocation policies are mainly categorised as pre-emptive and non-preemptive. Pre-emptive policies replace packets which are already in the switch buffers and non-preemptive policies, also known as threshold-based policies, prevent packets from entering the switch buffers. Threshold-based policies are further classified as static, dynamic, and adaptive control. Simulation of Markov state models for static threshold policies is used to analyse and deduce the distinctive characteristics of these policies.

Chapter five presents an improved dynamic buffer allocation policy, decay function threshold (DFT) policy. Two versions of this policy are derived and described. This policy is inspired from the exponential decay of sharable buffers as the input traffic intensifies on each output port in a shared buffer switch. Transient behaviour of DFT

policy is studied through transient analysis. Also, the ability of DFT policy supporting priority traffic by means of per-queue thresholds is discussed in this chapter.

Chapter six describes the simulation model developed to evaluate the performance of proposed buffer allocation policy. In this chapter, an overview of the development platform, parallel virtual machine (PVM), and a detailed description of the system model are provided. Note that a simulation program was developed by using the C programming language based on the model described in this chapter. Traffic source models, types of traffic (i.e. best-effort, multicast and QoS) and parameters used for performance evaluation are also discussed. Benchmarking results for developed simulation program is given in Appendix B.

Chapter seven presents the results obtained from the simulations. In this chapter, dynamic allocation policies, dynamic threshold (DT), dynamic queue threshold (DQT), partial sharing partial partitioning (PSPP) and decay function threshold (DFT) policies are evaluated and compared. Superiority of DFT over well-known DT in case of multicast traffic, and performance characteristics of DFT in case of QoS are studied.

Chapter eight finalises the discussions in this thesis and suggests directions for further research. An adaptive learning control scheme is proposed and preliminary results for this work are provided in Appendix C.

1.4. Original Contributions

During the course of this work the author has:

- Investigated logical buffering schemes (Chapter two). These are output queueing, input queueing, virtual output queueing and shared buffering.
- Investigated switch architectures with optical delay line (ODL) buffers (Chapter two). These architectures are categorised according to the positions of ODL buffers: single-stage feedforward, single-stage feedback, multistage feedforward and multistage feedback.
- Simulated and evaluated multistage interconnection networks (MINs) with buffered node architectures (Chapter three). Different types of buffering schemes including switch architectures with ODLs used as node architectures in MINs, simulated and evaluated under general traffic conditions. It has been shown that scalable all-optical switch architectures can be achieved from modular designs.
- Analysed the limiting features of the shared buffer switch architecture in the presence of imbalanced input traffic, and carried out a numerical analysis on existing buffer management policies in the literature (Chapter four). These policies are categorised as static threshold, dynamic threshold, preemptive and adaptive control.
- Proposed an improved dynamic threshold policy, decay function threshold (DFT), for shared buffer switch architecture (Chapter five). Proposed policy has been simulated (Chapter six) and evaluated under various traffic conditions (Chapter seven). Proposed policy provides an improved switch performance under multicast traffic.

- Proposed an integration framework for buffer management and bandwidth scheduling for the shared buffer switch architecture (Chapter five). This framework has been simulated (Chapter six) and evaluated under QoS traffic (Chapter seven). Proposed scheme employs lightly coupled buffer management and scheduling schemes within the same framework by means of employing per-queue thresholds for each priority class/queue.

1.5. List of Publications

1. B. Gazi and Z. Ghassemlooy, "Dynamic Buffer Management using Per-Queue Thresholds," Accepted for Publication in International Journal of Communication Systems.
2. B. Gazi and Z. Ghassemlooy, "Performance Comparison of VOQ Selection Policies in Scalable Packet Switches," Proceedings of the 5th International Symposium on Communication Systems, Networks and DSP (CSNDSP), Patras, Greece, 19-21 July 2006, pp. 319-323.
3. B. Gazi and Z. Ghassemlooy, "Threshold based dynamic buffer management: an overview," Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK, 27-28 June 2005, pp. 61-66.
4. R. J. Allarton, B. Gazi and Z. Ghassemlooy, "Comparative performance of 2×2 optical contention resolution topologies with multicast capability," Proceedings of PREP 2005, Lancaster University, UK, 30 March – 1 April 2005, pp. 98-99.

5. B. Gazi and Z. Ghassemlooy, "Exponential decay queue thresholds for optimal buffer sharing," The IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2005), Twenty-Third IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria, 15-17 February 2005, pp. 590-594.
6. B. Gazi and Z. Ghassemlooy, "Performance characteristics of cascaded delay-line node architectures in single-path interconnection networks," International Journal of Communication Systems, vol. 17, no. 5, pp. 447-459, 2004.
7. B. Gazi and Z. Ghassemlooy, "Sigmoid function based dynamic threshold scheme for shared-buffer switches," Proceedings of the 4th International Symposium on Communication Systems, Networks and DSP (CSNDSP), Newcastle, UK, 20-22 July 2004, pp. 424-427.
8. B. Gazi, Z. Ghassemlooy and G. Swift, "Cascaded optical delay line architectures in the performance of optical MIN," The 4th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'03), Chengdu, PR China, August 2003, pp. 277-281.
9. B. Gazi, Z. Ghassemlooy and G. Swift, "Optical multistage interconnection networks employing optical buffering," SET for Britain, House of Commons, London, UK, 17 March 2003, pp. 26, poster L1-27.

CHAPTER 2
AN OVERVIEW OF BUFFERING STRATEGIES IN PACKET SWITCHES

2.1. Introduction

In a packet switched network, each packet must traverse through a number of switching nodes to reach its destination. In a non-blocking switch fabric, two or more packets arriving simultaneously at switch inputs may be destined to the same output port, thus resulting in packet contention. Only one packet is allowed to leave the switch from that particular output port due to limited output port bandwidth and the rest of the packets are dropped, thus resulting in packet losses. Packet losses can be prevented by storing packets temporarily for later departure until contentions are resolved.

Buffering in high-performance packet switches is discussed in this chapter. An overview of the literature survey is graphically represented in Fig. 2.1. Buffering approaches can be studied in two domains: electronic and optical. In electronic domain packets are represented by using digital signals stored in random access memory (RAM) buffers. This type of packet storage provides flexibility when writing/reading to/from buffers. In optical domain, switching, buffering and header processing are done in optical domain without having to convert from optical to electrical and vice versa. Optical buffers are in the form of optical delay lines, as optical memories are not quite mature yet. Unlike electronic domain, optical buffing offers fast packet switching, but suffers from signal degradation due to continuous circulation of packets within delay lines. The rest of the chapter is organised as

follows. The next section provides an overview electronic buffering schemes and Section 2.3 describes packet switches that employ optical delay line buffers.

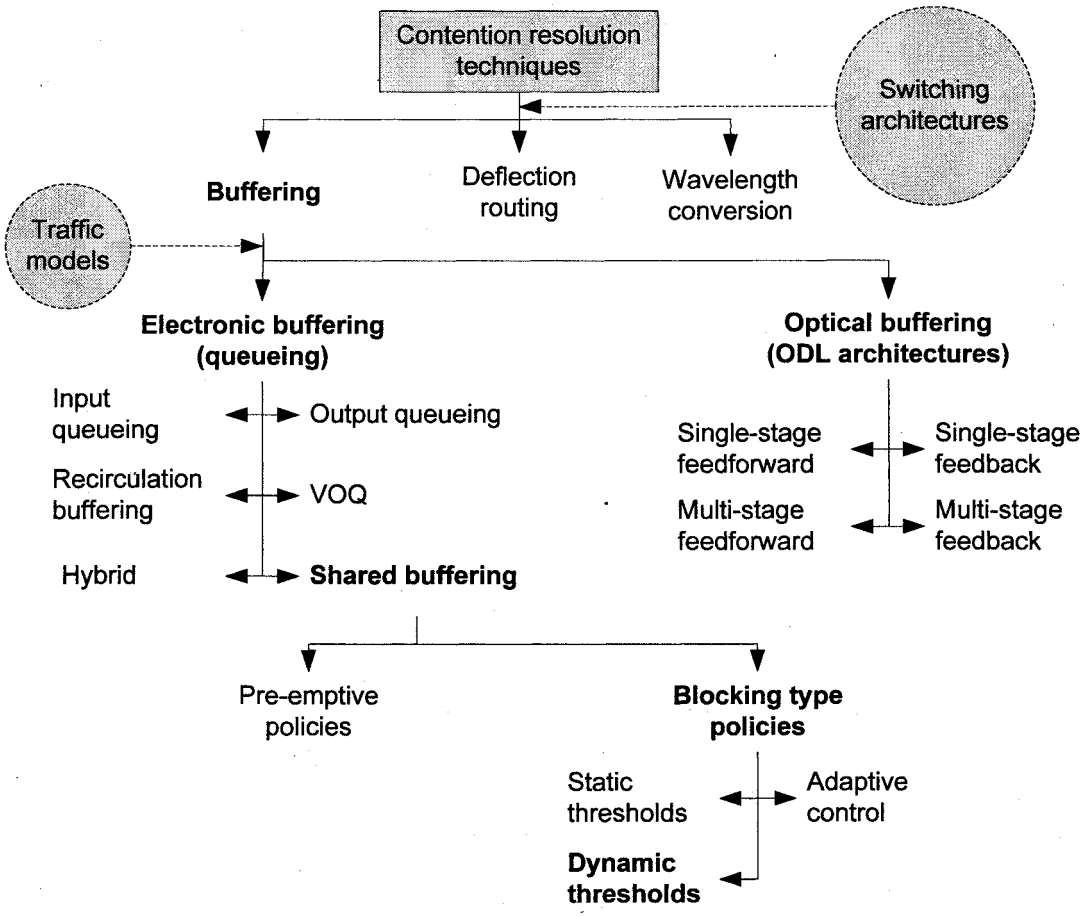


Fig. 2.1 – Literature survey at a glance

2.2. Buffered Packet Switch Architectures

Buffering approaches for high-performance packet switching can be essentially classified into four main categories according to the position of buffers: input queueing, output queueing, recirculation buffering and shared buffering.

2.2.1. Input queueing

In input queueing, a separate queue is placed on each input port of the non-blocking type switch fabric (Fig. 2.2). A packet arriving at an input port first enters the first-in-first-out (FIFO) buffer where it awaits access to the switch fabric. Packets remain in the input queues for the next scheduled departure for two reasons: (1) The buffer contains one or more packets waiting for departure using the same output port, or (2) there is a packet at the head of line (HOL) of an input queue which is in contention with other packets in HOL of other input queues. At the beginning of every time-slot, only the head of line packets try to gain access to the switch outputs. If k packets at the head of the input queues are addressed to different output ports, no contention occurs and all k packets are allowed to exit from the output ports of the switch. However, if all k packets are addressed to a particular output port, only one packet is allowed to use that output port with $1/k$ probability and other k-1 packets must wait until the next time-slot. New packets arriving at the input ports are blocked together with other packets behind the head of the queues from reaching output ports. As a result output ports are underutilised, known as HOL blocking (Hluchyj and Karol 1988).

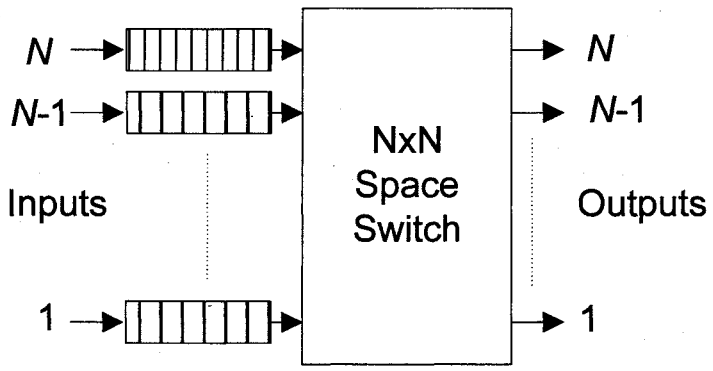


Fig. 2.2 – Input queueing

Studies in (Hluchyj and Karol 1988) have shown that for a large switch size ($N \rightarrow \infty$), throughput of a switch with input buffers is as low as $(2-\sqrt{2}) \approx 0.586$ under uniform independent identical Bernoulli arrivals in a slotted network. For instance, if k packets (at the HOL of the input queues) are destined for the same output port i, the switch controller picks one of the k packets at random with probability of $1/k$ and the other k-1 packets wait for the next time-slot forming a *HOL queue* for output port i. A HOL queue consists of packets at the HOL of input queues which are destined to the same output port, see Fig. 2.3. When input buffers are saturated, there are always packets waiting in the buffers. In each time-slot, at most one packet is sent from each non-empty input queue and on average $N.\rho_0$ packets depart from the switch in steady-state, where N is the number of output ports and ρ_0 is the switch throughput. Ideally the same number of packets arrive at the HOLs of the queues to replace departing packets. The distribution of the number of arriving packets to free input queues has the *Binomial* probabilities. As $N \rightarrow \infty$, this distribution becomes *Poisson* with rate ρ_0 and each *HOL queue* behaves like a discrete time M/D/1 queue (Karol et al. 1987). The mean steady state delay for an M/D/1 queue is given by Pollaczek-Khinchin mean formula (see Appendix A) as:

$$\overline{W}_i = p_0 + \frac{p_0^2}{2.(1-p_0)}. \tag{2.1}$$

At most N virtual queues, characterised as M/D/1 queues, are formed for each output port (i.e. HOL queues). In a heavily loaded system, none of the input buffers is empty; thus HOL *queues* together always fill the HOL of input queues. Hence the mean queue lengths of each *HOL queue* equals to 1 when $N = \infty$ and this is given as:

$$p_0 + \frac{p_0^2}{2.(1-p_0)} = 1 \quad \text{and} \quad p_0 = 2-\sqrt{2} \cong 0.586. \tag{2.2}$$

There are a number of ways for overcoming the head of line blocking property of input queueing. These are; dropping packets from HOL queues (Karol et al. 1987), look ahead contention resolution scheme (Thomas and Man 1993), output expansion, switch speed-up, input port expansion, and virtual output queueing (McKeown 1999). The rest of this section discusses these techniques.

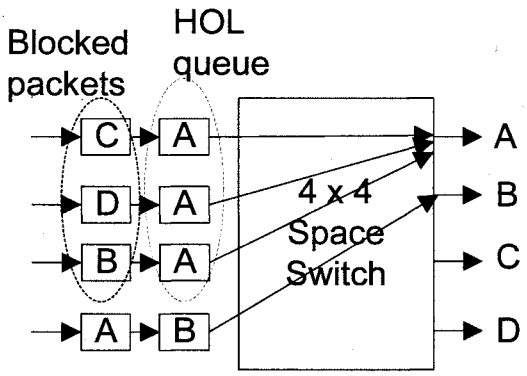


Fig. 2.3 – HOL queue

2.2.1.1. dropping packets from HOL queues

Whenever k HOL packets are destined for a particular output port in a time-slot, only one packet can be transmitted over the output port and the rest of k-1 packets wait in their queues for the next time-slot. A packet blocked behind one of k-1 packets has to wait at least one time-slot for departure, see Fig. 2.3. In order to prevent this, instead of storing k-1 packets in input queues, they are just dropped in order to allow blocked packets to move forward. However, dropping packets under small input loads reduces the switch throughput. This is because new packets do not arrive fast enough to replace dropped packets. In other words, there has to be a threshold to determine at which input load levels and switch sizes dropping policy offers a better performance than input queueing with no control.

Let $A_m(i)$ be the number of packets that are destined for the output i in mth time-slot, which has the Binomial probabilities (where p is the traffic load) and is given as (Karol et al. 1987):

$$\Pr[A_m(i) = k] = \binom{N}{k}(p/N)^k(1-p/N)^{N-k}$$

$$k=0, 1, \ldots, N. \qquad (2.3)$$

Let $I_m(i)$ represent the presence ($I_m(i) = 1$) or absence ($I_m(i) = 0$) of an output port i packet in mth time-slot. The arrival of packets at the output ports is independent and identically distributed (i.i.d.). The probability of a packet leaving output port i in mth time-slot is given by:

$$\Pr[I_m(i) = 1] = 1 - \Pr[A_m(i) = k] \qquad \text{where } k = 0 \qquad (2.4)$$

Therefore, the probability of port i receiving a packet in mth time-slot is given as:

$$\Pr[I_m(i) = 1] = 1 - (1 - p/N)^N \qquad (2.5)$$

Overall utilisation of switch outputs is given by:

$$p_0 = 1 - (1 - p/N)^N \qquad (2.6)$$

$$p_0 = 1 - e^{-p} \quad \text{for} \quad N \to \infty \qquad (2.7)$$

For small input loads the proposed solution offers low throughput performance. For instance, to achieve throughput greater than $2-\sqrt{2}$ when $N \to \infty$, the proposed scheme is the most effective when input load is $p > \ln(1+\sqrt{2})$. Note that this load threshold has to be calculated for different switch sizes by using Equation 2.6.

2.2.1.2. look-ahead contention resolution scheme (windowing technique)

The throughput in an input-queued switch can be increased by relaxing the strict FIFO queueing discipline of the input queues (Hluchyj and Karol 1988). Each input sends at most one packet, but not necessarily the HOL packet in its queue. Those contending packets are tested for a contention-free configuration in one time-slot in a window of w cycles. In the first cycle, HOL packets are allowed to contend for the outputs. At the end of the first cycle, if there are any unclaimed outputs due to HOL blocking, then in the second cycle the second packets of the input queues that have lost contention in the previous cycle contend for the remaining output ports. This process is repeated until all output ports receive a packet within the window of w. At the end of w cycles, if there are unclaimed output ports, these ports receive no packets. One important feature of this scheme is that the arbitration speed must be w times the line speed of the switch. When $w = 1$, this scheme is equivalent to input queueing with FIFO discipline.

An alternative approach to the above windowing scheme is maximising the number of assigned output ports in each time-slot, even allowing multiple selections from the same queue (Thomas and Man 1993). This scheme aims at 100% utilisation of the output ports. In the first cycle of the window, the algorithm starts filling in the unallocated output ports starting from HOL of the first input queue through Nth input queue. If there are any unallocated output ports, it continues with the second cycle starting from the first input queue. When all the outputs are assigned with a packet, the algorithm stops searching the window. Consider the following example (Fig. 2.4(a)), in a switch of size $N = 4$ with window size $w = 3$.

	Window		
Cycle	3	2	1
Queue 0	b	c	b
Queue 1	d	d	c
Queue 2	a	b	a
Queue 3	a	c	c

(a)

Output Ports	
*	**a**
*	**b**
*	**c**
*	**d**

	Window		
Cycle	3	2	1
Queue 0	b	c	-
Queue 1	d	-	-
Queue 2	a	b	-
Queue 3	a	c	c

(b)

Output Ports	
a	**a**
b	**b**
c	**c**
d	**d**

	Window		
Cycle	3	2	1
Queue 0	b	c	c
Queue 1	d	-	-
Queue 2	a	b	-
Queue 3	a	c	-

(c)

Output Ports	
*	**a**
*	**b**
*	**c**
*	**d**

	Window		
Cycle	3	2	1
Queue 0	-	b	c
Queue 1	-	d	c
Queue 2	-	a	b
Queue 3	-	a	c

(d)

Output Ports	
*	**a**
*	**b**
*	**c**
*	**d**

Fig. 2.4 – Improved windowing technique:
(a) Step 1, (b) Step 2, (c) Step 3, and (d) Step 4

In the first cycle packets *b*, *c*, *a* are allocated to their corresponding output ports (Fig. 2.4(b)). The HOL in queue 3 contends with HOL in queue 1 and therefore it is not selected. The algorithm starts searching the second cycle of the window. In second cycle in queue 1, *d* is not selected up to this stage, hence it is picked. At this stage the algorithm stops searching the window, as all ports are allocated and the next step is to rearrange the queues. Packets are not moved to the head of their queues to fill in empty gaps, as this is not fair for packets which have lost contention in the previous time-slots (e.g. packet *c* in queue 3). Rather packets are organised starting from HOL of queue 0. The HOL packet in queue 3 is moved to HOL of queue 0 (Fig. 2.4(c)). The packet in the second cycle of queue 0 is moved to HOL of queue 1. The packet in the second cycle of queue 2 is moved to HOL of queue 3 and all the gaps are filled in the same way (Fig. 2.4(d)).

This way of rearrangement resolves fairness issues among all packets within the buffer. However, unfairness in individual input queues is still a problem, because more than one packet can be picked from a queue. This can be simply solved by randomising the arriving packets over N input queues. The results provided in (Thomas and Man 1993) show that this windowing technique is better than the windowing scheme described in (Karol et al. 1987) for larger switch sizes.

2.2.1.3. hardware strategies for overcoming the HOL blocking

If output ports could allocate more than one packet in a single time-slot, then it is possible to reduce contentions among those packets destined for the same output. The output expansion is based on the channel grouping concept (i.e. grouping of separate physical channels). In an $N \times Nr$ switch, r packets can simultaneously access the same output port, see Fig. 2.5(a). However, only one packet in a time-slot can be cleared from an input queue.

In a packet switch where both switch fabric and output port channels operate v times faster than the input lines (see Fig. 2.5(b)); the input load on the switch is reduced by a factor of v. Unlike output expansion technique switch speed-up technique allows more than one packet to leave the same input queue in the same time-slot. Therefore, it is possible to state that speed-up technique is better than the output expansion technique when output expansion factor is equal to the speed-up factor.

Each input port can be expanded into s ports each feeding the switch through s sub-ports (see Fig. 2.5(c)). By this way, up to s packets from each input queue can be presented for contention. It is possible to clear more than one packet from each input

queue. Thus, one would expect this technique to have a higher throughput than the windowing techniques proposed in (Karol et al. 1987) and (Thomas and Man 1993).

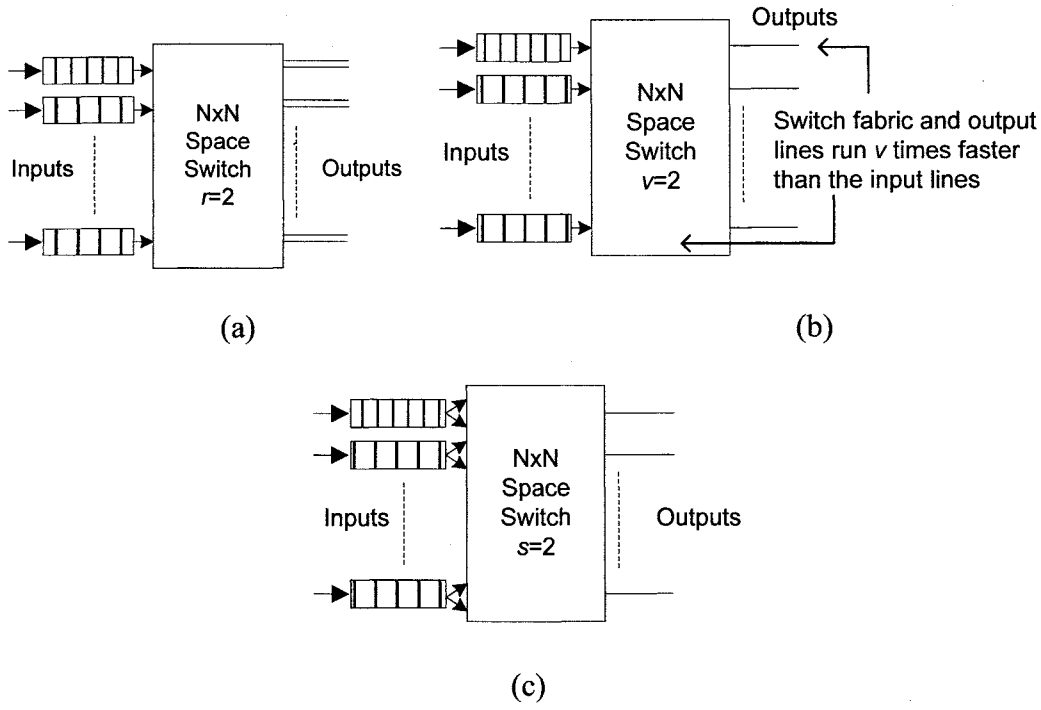


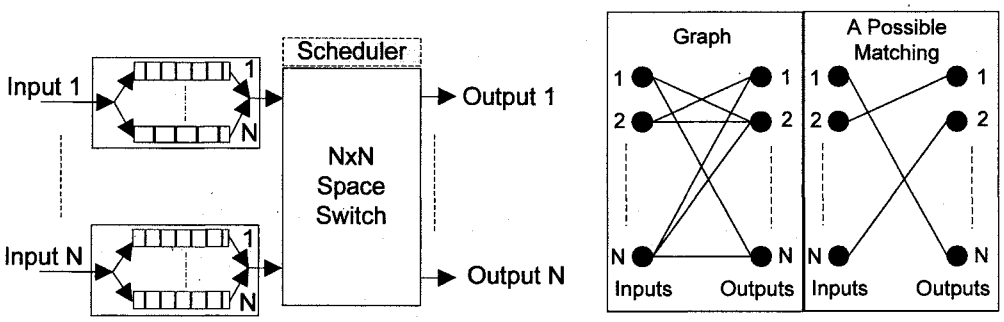
Fig. 2.5 – Hardware strategies for overcoming the HOL blocking: (a) output expansion, (b) switch speed-up, and (c) input expansion

2.2.2. Virtual output queueing

Virtual output queueing (VOQ) was first proposed to overcome the HOL blocking problem of input queueing (McKeown 1995). Approaches discussed in the previous section solve the HOL blocking problem and offer improved performances, but due to non-FIFO nature of these schemes packets destined for the same output port are treated unfairly. In VOQ, on arrival packets are sorted into N FIFO queues each destined to a specific output port (i.e. destination queueing) to preserve the FIFO order of arrivals. As depicted in Fig. 2.6(a), each input queue is divided into N sub-queues (e.g. one FIFO queue per output port). Therefore, in total N^2 queues are

required in an $N \times N$ non-blocking switch. As arriving packets are sorted into corresponding queues, a packet cannot be held up by another packet which is destined for another output port. As a consequence, HOL blocking is eliminated.

Virtual output queueing switch architecture consists of a sorter at each input port and a global scheduler. Sorter is responsible from sorting packets according to their destination addresses. On the other hand, scheduler aims to find a contention-free match between inputs and outputs of the switch to maximise the number of packets delivered at the end of each time-slot. This problem can be modelled as bipartite graph matching problem (see Fig. 2.6(b)). The main dilemma with this approach is that the maximum number of cells per time slot is aimed to be delivered, but this can take very long to compute and results in potential starvation of some input flows (McKeown 1999).



(a) (b)

Fig. 2.6 – Virtual output queueing: (a) switch architecture, and (b) bipartite graph matching

Various scheduling algorithms, such as maximum size matching (MSM), maximum weight matching (MWM), parallel iterative matching (PIM), iterative round robin matching with slip (*i*SLIP), iterative least recently used (*i*LRU), iterative longest queue first (*i*LQF), iterative oldest cell first (*i*OCF), reservation with preemption and

acknowledgment (RPA), have been developed to provide valid configurations with high throughput whilst keeping complexity low (see Table 2.1 for an overview).

Table 2.1 – An overview of the common scheduling algorithms

Scheduling Algorithm	**Running-time complexity to converge maximal matching (*N*x*N* switch)**	**Description**
MSM	$O(N^{5/2})$	A general class of scheduling algorithm. Finds instantaneous maximum number of matching between inputs and outputs. Can lead to starvation of an input-output flow under certain traffic patterns. Achieves 100% throughput.
MWM	$O(N^3)$	A general class of scheduling algorithm Aims to maximise the total weight assigned to edges between vertices in a bipartite graph. Delivers 100% throughput. Provides low delays by keeping queue sizes small. Too complex to implement. Achieves 100% throughput.
PIM (McKeown and Anderson 1998)	$O(N.\log N)$	Uses *randomness* to avoid starvation and to converge to a maximal size matching with less number of iterations. Randomness feature of PIM is difficult and expensive to implement and can lead to unfairness between flows. It performs poorly for a single iteration (2×2 switch) and limit throughput to 63%.
*i*SLIP (McKeown and Anderson 1998)	$O(N.\log N)$	Belongs to MSM class. Similar to PIM, but replaces the randomisers with round robin arbitrers. Round robin arbitrers provide fairness among flows. It has a better performance than PIM for only a single iteration and also has lower implementation cost. It was used in *Tiny Tera* router (McKeown 1999). Running time complexity can be reduced by adding parallelism or pipelining (McKeown 1999).
LQF (McKeown 1995)	$O(N^2.\log N)$	Belongs to MWM class. Those queues that are heavily occupied are served first. Can lead to indefinite starvation of one or more inputs. Complex to implement.
OCF (McKeown 1995)	$O(N^2.\log N)$	Belongs to MWM class. It gives service priority to those cells that have been queued for a long time. It is too complex to implement.
*i*LQF (McKeown 1995)	$O(N.\log N)$	Belongs to MWM class. Iterative version of LQF. Suffers from starvation as in LQF. Exhibits stable performance under non-uniform traffic.
*i*OCF (McKeown 1995)	$O(N.\log N)$	Belongs to MWM class. Iterative version of OCF. Those packets in the head of their queues with higher waiting times are served first. Like *i*LQF, it is stable under non-uniform traffic.
RPA (Marsan et al. 1999)	$O(N^2)$	Belongs to MWM class. The switch input ports can indicate their most urgent cell transfer needs and an acknowledgment round to allow input ports to determine what cell they can actually transfer toward the desired switch output port.

VOQ architecture is proposed for cell-based networks, such as ATM. For such high performance systems, the following properties are desirable in a scheduling algorithm: high throughput, starvation-free, and simple to implement. The algorithm should resolve bottlenecks in the switch/router by providing 100% utilisation at the output ports. Also, it is important that the algorithm performs fast matching to allow cells to be moved forward. A scheduling algorithm must have low complexity to perform fast matching between inputs and outputs to avoid backlog of requests. Otherwise, algorithm performance bottleneck results in low throughputs and long packet delays. Some of the algorithms discussed in Table 2.1, such as *i*SLIP, can be implemented by using a parallel approach to reduce the computational complexity.

2.2.3. Output queueing

In this buffering scheme, fixed length FIFO queues are situated at the outputs of $N \times N$ non-blocking switch (Fig. 2.7). Packets destined for the same output port are queued at the switch outputs. Unlike input queueing, output queues only store those packets which are destined for the same output port. Therefore, this type of packet buffering does not suffer from HOL blocking. Besides packet contentions is reduced to a minimum and only occur at output ports. Nevertheless contentions are unavoidable due to statistical nature of arrivals (Hluchyj and Karol 1988). Packet losses occur when there is no sufficient buffer space to accommodate those packets destined for the same output port. This type of buffering is work-conserving, as there is always at least one packet at switch outputs waiting to be served. This buffering scheme achieves optimal throughput-delay performance. Packets are delayed for a minimum amount of time due to congestion caused by packets that arrive simultaneously at different input ports and destined for the same output port.

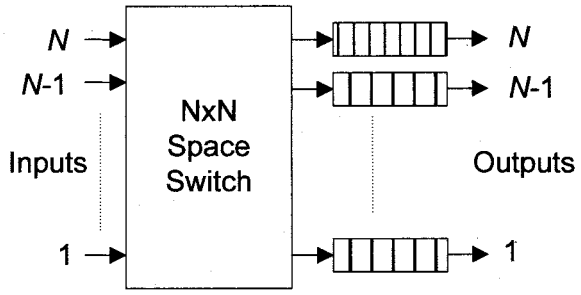


Fig. 2.7 – Output queueing

It is assumed that all N queues behave in a similar way and the switch performance can be obtained by symmetry, i.e. by observing a particular output queue. Therefore the mean waiting time for output queueing is modelled as M/D/1 queue when $N = \infty$ and queue size of a single output queue is infinite (Karol et al. 1987). This is given as a function of offered load p:

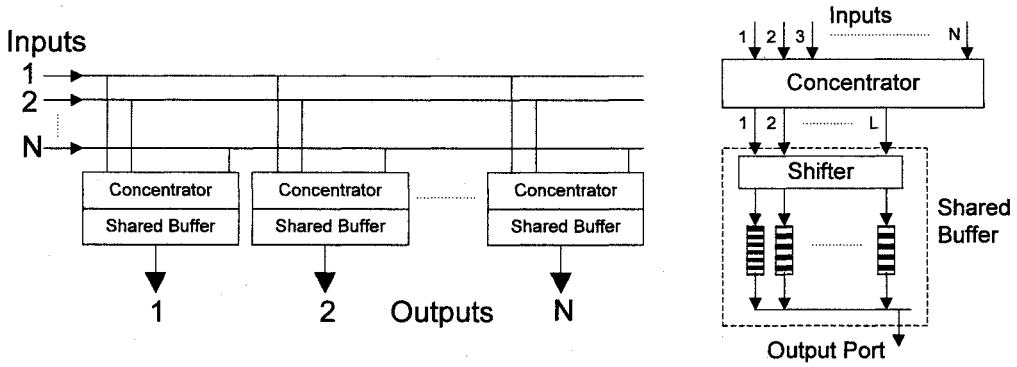
$$\overline{W} = \frac{p}{2.(1-p)} \tag{2.8}$$

One important feature of this type buffering is that memory write speed at output queues must be N times the line speed. For example, if k packets arrive in the same time-slot at the inputs of an $N \times N$ switch and are destined to the same output port, all k packets must be admitted to the same output queue simultaneously within the same time-slot. Only one packet leaves the output queue in a time-slot and the rest of packets must wait in the queue. For instance, B is the bandwidth of a single port and the exact internal bandwidth requirement for simultaneous read/write for a single output port is $(k + 1).B$ (i.e. k packets arrive and one packet leave in the same time-slot). For this reason, it is difficult to achieve large switch sizes with high bandwidth input/output ports, i.e. scalability handicap, whereas input queueing does not suffer from such drawback due to immediate admittance of packets on arrival. There are

number of ways of solving this problem without having to employ memory or switch fabric speed-up: concentrator (Yeh et al. 1987), N lines connection to each output queue (Barri and Goubert 1997), crosspoint buffers (Nojima et al. 1987), and output bandwidth expansion. These techniques are discussed below.

2.2.3.1. concentrator (the knockout switch)

Concentrator was first proposed for the knockout switch to eliminate the speed-up problem of output buffering. The knockout switch is an $N \times N$ switch which is based on output buffering. As shown in Fig. 2.8, there are three main parts of this switch architecture. These are interconnection fabric, concentrator, and buffers.



(a) (b)

Fig. 2.8 – The knockout switch: (a) interconnection fabric, and (b) shifter and buffer

All packets destined for the same output port enter the concentrator of that particular port (see Fig. 2.8(b)). Concentrator performs N to L concentration by eliminating $N - L$ packets, where $N > L$. That is, k packets enter the concentrator and if $k \leq L$, all k packets enter the buffers. However, if $k > L$, all L outputs of the concentrator will have at least one packet and the rest $k - L$ packets will be dropped by the concentrator. Packet loss probability is admissible for very large L. For example, when $N = \infty$ and

input load is 1.0, the packet loss rates of the concentrator for $L = 6$ and $L = 10$ are 10^{-5} and 10^{-8}, respectively (Yeh et al. 1987).

As illustrated in Fig. 2.8(b), when L packets arrive at the buffers, they are enqueued at the end of L queues through a shifter. The shifter has L states and performs circular right shift. In every time-slot, according to the state of the shifter, a different matching is formed between outputs of the concentrator and inputs of the shared buffer. If buffers were to be loaded directly from the concentrator, the left most buffers would fill up quickly and might overflow. The shifter prevents this by distributing batch arrivals by simply shifting to the right. For instance, if the last FIFO queue that received a packet is j, then the next k packets will be distributed those buffers from $j + 1$ to $j + k$. The shared buffer allows complete sharing among L FIFO queues. Queues are served by using a token mechanism starting from queue 1 to L. When a queue is served, it then passes the token to the next.

2.2.3.2. N lines connecting to each output buffer

In order to prevent switch fabric speed-up, N lines from the inputs can be connected to buffers at the output port buffers. That is, an output queue must have N lines connected to it in order to achieve the required speed-up. Although no packets are lost, this approach requires a large number of interconnections within the switch fabric with the speed-up problem of the buffers still remaining unresolved. This problem can only be solved by having N queues at the output ports for each line from input ports where queues are served by using a token mechanism as in the knockout switch (Hunter 1995).

2.2.3.3. crosspoint buffers

Crossbar switches with crosspoint buffers are internally non-blocking and simple to implement. This type of switches implements queueing function at the crosspoints of the crossbar switch (Fig. 2.9). The amount of packet allocation unit required for an $N \times N$ switch is $b.N^2$, where b is the amount of allocation units in a single buffer at each crosspoint (N^2 crosspoints). This feature of crosspoint buffering prevents scaling to large switch sizes (due to square growth). Nevertheless crosspoint buffers have simpler scheduling algorithms than unbuffered crossbar, as unbuffered crossbar must find a non-conflicting match between inputs and outputs (Chuang et al. 2005).

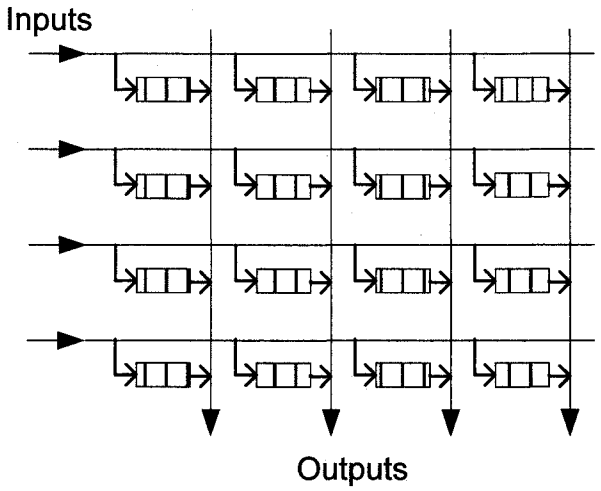


Fig. 2.9 – Crossbar switch with crosspoint buffers

2.2.3.4. output bandwidth expansion

Bandwidth expansion of output ports provides fast clearance of packets. Bandwidth expansions can be done in two possible ways: (1) extra physical channels can be added at each output port or (2) the speed of output channel is increased by providing higher bit rates or by using multiplexing techniques (e.g. wavelength multiplexing,

time-division multiplexing) (Hunter 1995). Bandwidth expansion results in inefficient utilisation of output port channels when the input load is relatively very small.

2.2.4. Shared buffering

In shared buffering, incoming packets are randomly pooled into a central memory area (e.g. random access memory) and read when required. A logical representation of this architecture is presented in Fig. 2.10. As in output queueing, there is a separate queue for each output port. However, the queue lengths are dynamic and memory allocation for each output port is on demand. Packets destined for the same output are logically linked in the form of a linked-list. Queue sizes can grow indefinitely, only if total allocated memory size does not exceed the pre-determined buffer memory size. The advantage of this approach is that it achieves optimal throughput-delay performance under uniform input traffic (Karol et al. 1987). Unlike output queueing, this performance is achieved with a requirement for smaller buffers. Another advantage is that complex buffer sharing and management disciplines, such as active queue management, can be implemented. Nevertheless, complexity of memory management is the main concern of this approach. As in output queueing, memory read/write speed has to be N times the line speed in an $N \times N$ switch.

Considering that occupied buffer space must not exceed predetermined buffer size, admitted packets must be selected through an arbitration mechanism. This is required to provide fairness among those packets that arrive simultaneously. For instance, l packets request buffer space from the shared memory and there are k available packet slots in the memory. When $l \leq k$, all packets are written into the memory without having to drop any packets. When $l > k$, the buffer management unit has to decide

which packets will be admitted. The first k packets are selected at random and $l - k$ packets are rejected from entering the switch buffers. As well as calculating buffer occupancy in terms of number of fixed-length packets, this can also be calculated as the sum of number of bytes in memory (in case of variable-length packets). In such a case, packet admission depends on whether the packet can be fitted into available memory space. If the packet size is bigger than empty buffer size, this packet is rejected. This way of memory management is rather complicated when compared with fixed-length packet case. Besides, memory might be poorly utilised.

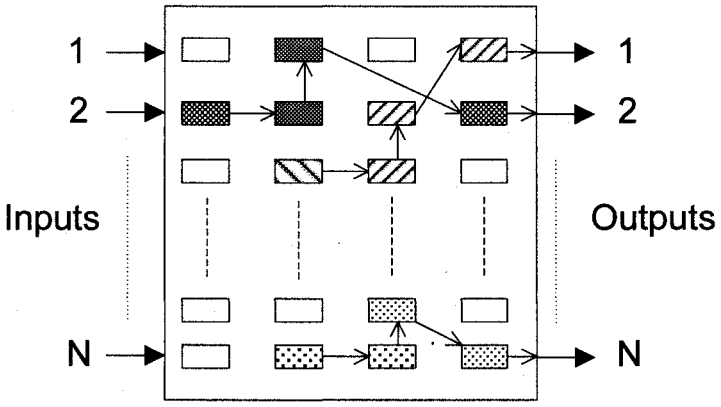


Fig. 2.10 – Shared buffering

The main drawback of this buffering scheme is that a heavily loaded output port could affect other ports. A heavily loaded output port could dominate the usage of the shared resource and drain rest of the output ports (Irland 1978; Wei et al. 1991). An in-depth study will be carried out later in this thesis to observe the performance characteristics of the shared buffer switch and possible approaches for overcoming the performance degradation due to imbalanced bursty network traffic.

Despite the requirement for memory management and memory read/write speed-up, the shared buffer switch achieves optimal throughput-delay ratio and better resource

allocation (Hluchyj and Karol 1988). Note that shared buffer switch architecture is ideal for supporting multicast traffic. For these reasons it attracted attention of researchers and engineers. Examples of this type of switch architecture include:

- Hitachi switch (Kozaki et al. 1991): Shared buffer ATM switch with a port speed of 155.52 Mbps.
- IBM Vulcan switch (Stunkel et al. 1995): High-performance switch, which is a low-latency, high-bandwidth switching interconnect that binds together RlSC System/6000 processors.
- IBM Prizma switch (Engbersen 2003): IBM Prizma pioneered the separation of control and data paths in its internal architecture. Packet payload is routed through the data path only. Prizma switch consists of parallel input-output routing paths that allow simultaneous read/write from/to the shared buffer.

2.2.5. Recirculation buffering

Recirculation buffering consists of a pool of memory completely shared among output ports; also known as completely shared buffering (Fig. 2.11). The buffer module consists of inputs, which are fed from the non-blocking space switch, and outputs, which are inputs back to the space switch. There are $N.b$ inputs and outputs of the buffer module. Hence, switch size is scaled from $N \times N$ to $(N + N.b) \times (N + N.b)$, where $N.b$ lines connect to single time-slot delay buffer module and N lines are connected to the outputs of the switch. For instance, if there are k packets addressed to the same output port (e.g. output port i), where $k = 1, 2,..., (N + N.b)$, only one packet leaves the switch fabric from output i and the rest of k-1 packets are sent to $N.b$ inputs of the buffer module where they are delayed one time-slot and fed back to the switch

fabric with other N newly arrived packets. After every recirculation, one packet leaves the group of k-1 packets and meanwhile newly arrived packets, those which are addressed to port i, join this group. Grouping of packets creates logical queues within the same physical memory. This technique was first proposed in STARLITE switch (Huang and Knauer 1984). Studies in (Hluchyj and Karol 1988) have shown that it achieves optimal throughput-delay performance as output queueing, but requires smaller buffer size to achieve a desired level of packet loss rate. Instead of complete partitioning, buffer space is allocated on-demand according to the volume of input traffic.

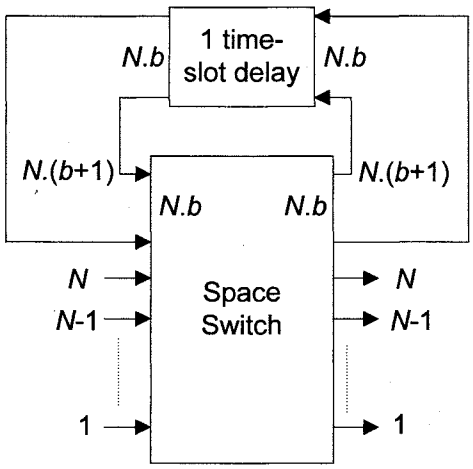
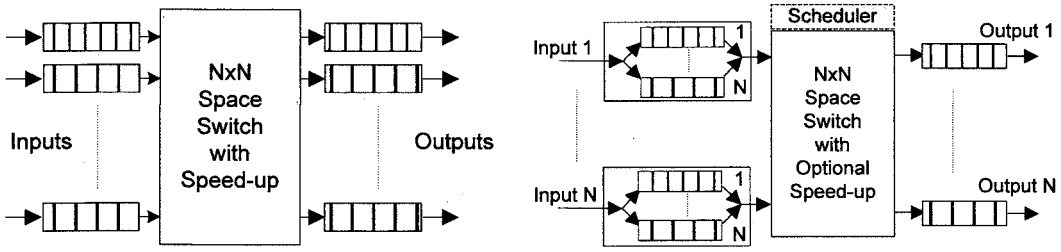


Fig. 2.11 – Recirculation buffering

Even though recirculation buffering achieves a good degree of performance, it requires large number of delays. For instance, 640 packet delays are required to achieve packet loss rate of 10^{-12} in a 64 × 64 switch under 0.9 traffic load (Hluchyj and Karol 1988). Recirculation buffering cannot be scaled to large switch sizes. Moreover it fails to deliver stable performance under imbalanced input traffic due to shared resource feature.

2.2.6. Hybrid: Combined input output queueing

Combined input output queueing (CIOQ) switch architecture uses buffer modules both at the inputs and outputs of a non-blocking packet switch. Early versions of this approach use FIFO queues at the inputs and recent approaches employ VOQ (Fig. 2.12).



(a) (b)

Fig. 2.12 – Combined input output queueing: (a) CIOQ with FIFO queues at inputs, and (b) CIOQ with VOQ

The CIOQ switch architecture with FIFO queues at the inputs, shown in Fig. 2.12(a), suffers from HOL blocking, which degrades switch throughput performance for large switch sizes. As mentioned earlier, switch speed-up improves the throughput performance. Besides, additional buffering may be required at the switch inputs due to buffer limitations at switch outputs. In (Iliadis and Denzel 1993), it was shown that a backpressure mechanism can be adopted to prevent overflow at the switch outputs. In his work, Iliadis analysed the performance implications of the CIOQ switch architecture with backpressure mechanism by using arbitrary buffer depths and large switch sizes under synchronous and asynchronous network models. It was also shown that performance close to output queueing could be achieved with appropriate switch speed-up and by employing a scheduling algorithm. In (Chuang et al. 1999), it was

shown that CIOQ employing FIFO queue with a speed-up of $2-1/N$ behaves identically as output queueing under uniform input traffic. With the expense of more complex scheduling algorithms, requirement for internal switch speed-up can be eliminated. Nevertheless, scheduling algorithms, such as critical cell first (CCF), are rather complex with running time complexity of at most O(N). This feature of CIOQ makes it unsuitable for fast packet switching.

In CIOQ with VOQ, Fig. 2.12(b), HOL line blocking of input queueing is resolved by employing VOQ with work-conserving graph matching algorithm. Besides, internal speed-up can be eliminated due to work-conservingness and low running complexity of the scheduling algorithm. In (Leonardi et al. 2001), it was suggested that having a speed-up factor of two is sufficient to reduce the complexity whilst leading to a performance close to output queueing. In this study it was proved that speed-up factor of two is stable for most of the scheduling algorithms.

The main motivation for developing CIOQ switch architecture has been matching the performance of output queueing by eliminating memory bandwidth speed-up requirement. On the other hand, scheduling algorithm complexity of VOQ can be reduced by simply increasing internal speed-up. By this way a trade-off can be achieved between scheduler complexity and switch fabric speed-up in order to match the switch performance of output queueing. Commercially available implementations of this architecture are the Sunshine switch, developed by AT&T Lab, (Giacopelli et al. 1991) and IBM Prizma switch-on-a-chip technology (Minkenberg and Engbersen 2000).

2.3. Optical Delay-line Switch Architectures

In order to resolve contentions in electrical packet switches, store-and-forward technique is used where contending packets are stored in a queue and accessed randomly or in FIFO order. This is only possible with random access memory (RAM). In optical delay-line switch architectures, optical random access memory is not readily available. For this reason, the only way to store packets is to use optical fibre delay lines (ODLs), where packets are delayed one or more packet length duration (see Fig. 2.13). Packets are only accessible at the end of fixed circulation time.

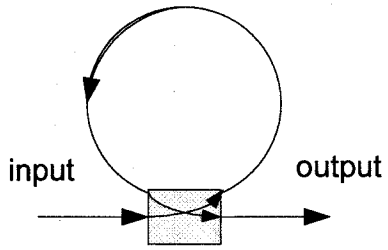


Fig. 2.13 – Optical delay-line

Packets, in the form of soliton pulses (a special kind of wave that can travel for a long distance, maintaining its shape and velocity without dispersing its energy) enter the recirculation optical fibre loop and spend a discrete amount of time until they are switched out. The number of bits K that can be stored in a fibre of length L is given as (Als and Ghassemlooy 2004):

$$K = \frac{nL}{c} \times B \tag{2.9}$$

Where c and B denote the speed of light and bit rate respectively. Assume that a fibre delay line is implemented to store a single ATM cell which is 48 bytes of size. Note

that each soliton bit lasts 0.1 ns (i.e. at rate 10 Gb/s) and therefore required fibre length is ≈7.68 meters. The main drawback of optical fibre loops is that as packets travel over the fibre, optical signal power deteriorates due to noise and requires amplification. Hence, bit error rate (BER) increases with the number of recirculations in the ODL (Als and Ghassemlooy 2004). It is suggested that provided with higher input powers and signal amplification, packets can be stored longer.

The rest of this section provides an overview of switch architectures that employ ODL buffers, namely all-optical switch architectures. All-optical switch architectures process packet headers, store packets and switch packets in optical domain without having to convert from optical to electrical and/or vice versa. ODL switch architectures are fundamentally categorised as (i) single-stage feedforward, (ii) multi-stage feedforward, (iii) single-stage feedback, and (iv) multi-stage feedback.

2.3.1. Single-stage feedforward

Single-stage feedforward architectures, such as OASIS switch (Gabriagues and Jacob 1995), broadcast-and-select (Gambini et al. 1998), and the staggering switch (Haas 1992), consist of one stage of optical delay lines where packet contentions are resolved and are easier to control.

2.3.1.1. OASIS switch

OASIS switch emulates an output buffered switch (Hunter et al. 1998a). Each packet entering the optical buffer is delayed by the amount of time slots from 0 to b, where b is the depth of the buffer, and is directed to one of the available outputs. In order to emulate an output buffered switch, a counter is used to count the number of packets in

the buffer destined for that specific output port. In other words, a virtual FIFO queue is formed for each output port. When a packet enters the buffers, if it is destined for the output port i, the counter C_i for the output port i is incremented by one and the packet enters a delay length of C_i. To preserve the FIFO order of arrivals, if more than one packet is destined for the same output port, each packet undergoes successive delay lengths and minimum experienced delay is $C_i = 0$ when there is no packet in the virtual queue. When a packet leaves, the corresponding virtual queue counter is decremented by one. When $C_i = b$, all newly arrived packets are discarded, as buffers have encountered an overflow.

There are two ways of implementing the OASIS switch. The first one uses passive coupler with tunable wavelength converters (TWC). TWCs situated at each input port encode incoming packets according to the wavelength of the corresponding output port, and the passive coupler is used for the routing function. Hence packets destined for a particular output traverse through the switch by using the same wave-length. Similarly, the second approach uses arrayed wave guide (AWG) instead of passive couplers in order to reduce signal loss. An AWG is a device which can separate and combine signals with different wavelengths. OASIS implementation with AWG achieves a better BER at higher bit rates (Hunter et al. 1998a). In both hardware implementations of the OASIS switch, the algorithm described earlier is used to manage the virtual queues. Note that it is not possible to employ packet pre-emption and priority classes. OASIS approach exploits the same packet loss rate and delay as output-buffering, as it employs a similar queueing technique (i.e. a queue for each output port) (Hunter et al. 1998a).

2.3.1.2. broadcast-and-select switch (ACTS KEOPS project)

In broadcast-and-select switch, each packet entering from different input ports are encoded with fixed wavelength converters so that packets from different input ports have different wavelength signatures. A packet arriving from a particular input is encoded with a specific wavelength and broadcast to all ODLs by means of a star coupler (a passive optical coupler that has a number of input and output ports). Each output can then select a packet from one of the ODLs through semiconductor optical amplifier (SOA) switches (SOA is a device used for switching and amplification purposes). Each of the ODLs has different integer length delays from 0 to b, where the buffer depth is given as $M = \sum_{i=1}^{b} i$. If a packet is not selected in the current time slot, it is discarded. Discarded packets have copies in other delay lines waiting to be selected in the next time slot. As each packet is broadcasted to all delay lines with all possible delays and all output ports, it is possible to employ broadcast/multicast operation and packet priorities. Broadcast-and-select switch has the same performance as an output buffered switch. There are two SOAs on the path from inputs to outputs, which add additional noise, and also high bit rates introduce power penalty (dB) (Gambini et al. 1998).

2.3.1.3. staggering switch

The staggering switch consists of two non-blocking space switches, which are interconnected by delay lines of different lengths (Fig. 2.14). The first switch is an $N \times m$ non-blocking switch and used for scheduling newly arrived packets by inserting them into appropriate delay lines. The second switch is an $m \times N$ switch, which is used for routing packets to their destination ports. Scheduling stage is connected to

the switching stage by m lines, where ith line has the delay duration $d_i = i$ and $i = 0, 1, ..., m$. Note that each packet is of fixed length and delay lines are integer multiples of packet length in a slotted network. Hence, there are $m.(m+1)/2$ packet storage locations.

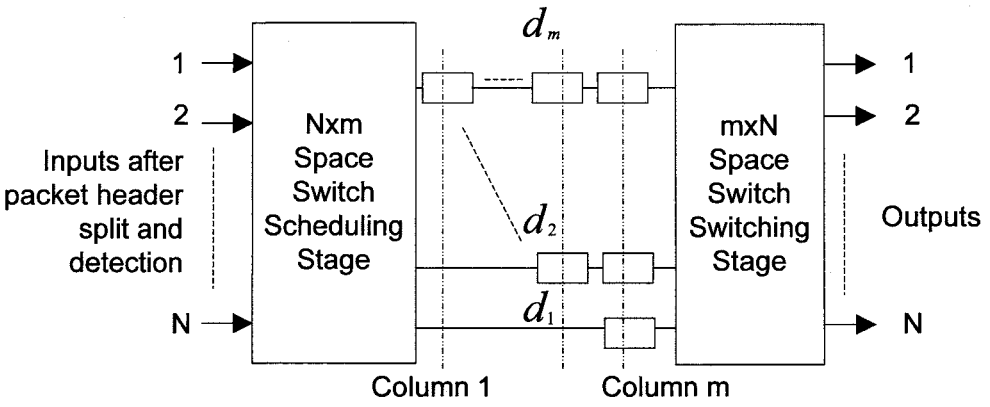


Fig. 2.14 – Staggering switch

The purpose of the scheduling stage is to arrange incoming packets in such a way that only one packet arrives at a particular output port in a given time-slot. Scheduling stage does this by delaying the contending packets by different amount of delay durations. There are a number of scheduling algorithms that can be used for scheduling (Haas 1992). One approach scans inputs sequentially and for each non-empty input port the algorithm tries to insert a packet in the lowest possible delay line. There are two restrictions for this insertion: (1) No other packet is inserted into this delay line in the same time-slot, and (2) there are no other packets in the same column destined for the same output port. By this way, all output ports can be utilised efficiently. This algorithm gives priority to those ports with smaller port numbers, as it scans inputs sequentially from 1 to N. Output ports with greater numbers are more likely to suffer from increased delays. For this reason, the above algorithm is slightly

modified to allow selection of input ports randomly, called sequential non-biased assignment (Haas 1992). Throughputs of the both approaches are the same under uniform uncorrelated input traffic, e.g. uniform Bernoulli. However, fairness among different flows is better fulfilled with the second approach.

Staggering switch provides a degree of buffer sharing among packets destined to different output ports, but this sharing is not like random access memory (Karol 1992). When a packet enters the delay line, it cannot be removed until it comes out from the other end of the delay line. Staggering switch cannot support priority traffic and buffer locations are not shared efficiently. One reason for this is that no two packets that are addressed to the same output port can be inserted in the same column. This location is left unoccupied, unless a packet with a different address is inserted. The scheduler is lack of an optimisation mechanism whose purpose is to minimise the number of wasted delay lines whilst ensuring no packet contentions take place at the output ports. When $m > N$, $m - N$ number of delay lines are wasted. Finally, the scheduler has to waste some delay line locations to preserve the FIFO order of the packets.

In (Karol 1992), it was shown that the staggering switch falls short of the performance offered by the shared buffering due to the way packets are stored in the delay lines (i.e. no two packets appear in the same column) and therefore it is rather characterised as output queueing. In (Karol 1992), it was shown that when the number of delay lines is equal to the switch size, $m = N$, the staggering switch outperforms the output queueing for relatively small switch sizes (e.g. $N = 8$ and $N = 16$).

2.3.2. Single-stage feedback

2.3.2.1. multiwavelength loop switch

In this buffered switch, multiple packets can be stored on the same ODL at different wavelengths so that packets do not contend once they are in the delay line. Each incoming packet is encoded by using TWC and placed onto an available wavelength. Each packet experiences one time slot delay. When a packet is to be removed from the buffer, the tuneable filters at the output ports are tuned to the original wavelength of that packet. Buffer management mechanism must keep a track of occupied wavelengths and those packets that are scheduled for departure in the next time slot. This switch emulates an output buffered switch and achieves the same packet loss probability and delay performances with a buffer depth equal to the maximum permissible number of circulations (Hunter et al. 1998a). The maximum permissible number of circulations is the maximum duration that a signal can be delayed before it fades due to power penalty introduced by noise.

This architecture is capable of employing packet preemption and priorities, as incoming packets can overtake those packets that are already waiting in the buffers. Besides, packets can be selected randomly from the buffer without having to conform to the arrival order of the packets.

2.3.2.2. SMOP (shared memory optical packet switch)

SMOP switch is an all-optical version of the Starlite switch (see Fig. 2.15). SMOP switch consists of N input/output fiber links, m circulation delay lines of integer multiple of packet lengths for storing packets, $(N+m)\times(N+m)$ non-blocking optical space switch and a control module to schedule and route packets. Unlike the

Starlite switch, buffer module is not one packet length delay. There are m delay lines with lengths d_1, d_2 ... and d_m, where $d_i = i$ and $i = 1, 2, \ldots, m$, and the total buffer size is given as $M = m.(m+1)/2$. The smallest unit length of a delay line is of one packet length. In every time slot, at most $N + m$ packets arrive at the switch and at most N packets are allowed to leave the switch; the rest of the packets are sent to the delay line buffers. A control mechanism determines which packets will be routed to the output ports and which ones will be stored. The control mechanism exploits the following characteristics (Karol 1993): (i) keeps track of all packets in the buffer by using a control table, (ii) keeps packets in their FIFO order, (iii) supports priority traffic, and (iv) ensures that the packets recirculate in delay lines only a small number of time in order to prevent signal degradation (e.g. less than ten fixed length packet delays).

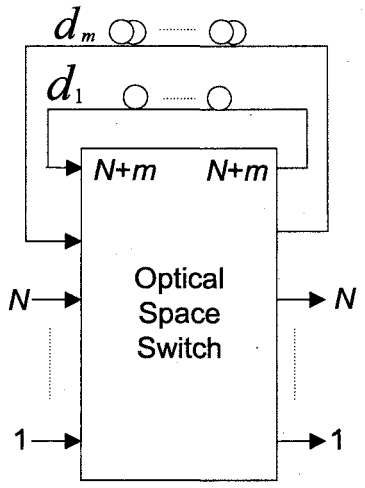


Fig. 2.15 – Shared memory optical packet switch

Two control schemes proposed in (Karol 1993) are non-FIFO and FIFO based algorithms:

A- **Non-FIFO algorithm:** At each time-slot:

(a) Start from longest delayed packets in the buffer towards the shortest delay and assign packets to the output ports.

(b) If there are empty output ports, then consider incoming packets.

(i) If there is no priority among incoming packets, then make a selection among incoming packets using a random fashion and send unselected packets to the buffer.

(ii) If there is a priority, then select the high priority packets, assign them to the empty output ports and send the rest of the packets to the buffer. (Packets are placed in the buffer according to their priority. High priority packets undergo shortest delays).

B- **FIFO Algorithm:** At each time-slot:

(a) Those scheduled packets arriving at the space switch are routed to their output ports.

(b) Any newly arrived packets at the input ports are allowed to exit from the output ports, if there are empty output ports after routing the scheduled packets. This does not violate FIFO order, as long as there are no scheduled packets for a particular output port.

(c) The remaining packets are sent to the buffer, trying to avoid more than one packet destined for the same output port in the same time-slot.

(d) While allocating packets to ODLs and routing packets to output ports, priority is given to those packets with higher number of recirculations.

In (Karol 1993), it was shown that SMOP achieves packet loss performance close to shared memory regardless of the control scheme used. Due to its feedback feature SMOP switch supports priority traffic. The main drawback of this architecture is that, as in Starlite switch, the increased dimension of the optical space switch causes scalability issues. For instance, small size SMOP switches can be used as a building block in a multistage architecture, wavelength division multiplexing (WDM) can be used to increase the capacity of the delay lines, or a modular design can be employed to support large switch sizes (Karol 1993).

2.3.3. Multi-stage feedforward

2.3.3.1. cascaded optical delay line (COD)

COD is a 2×2 buffered multistage switch architecture that uses smart crossbars and delay lines as building blocks. Smart crossbar is a 2×2 unbuffered photonic switch with cross and bar states. When two packets destined for the same output port arrive at the inputs of the smart crossbar, it uses the cross priority policy and sends one of the packets to its destination and deflects the other. As long as there is no contention, the packets are sent to their destination ports. TC (track changer) and TTC (twin track changer) modules are two versions that can be built up by using the smart crossbars (see Fig. 2.16(a)).

A cascaded organisation of a smart crossbar and n TC (or TTC) modules is called the baseline architecture. This architecture is depicted in Fig. 2.16(b). TC and TTC use the analogy of track changing trains (Cruz and Tsai 1996). Based on this analogy, a packet tries to get on the correct track before it exits from its destination output port. TC baseline architecture has three significant features. First of all, it is work

conserving. In other words, output ports are always utilised with undeflected packets, as long as there is at least one packet in ODL at some stage of the baseline architecture. Secondly, packets are lost (deflected) if and only if buffer overflow occurs. Thirdly, TC baseline architecture preserves the FIFO order among those packets destined for the same output port. TTC module consists of two delay lines. Thus, it offers twice capacity as the TC module leading to lower packet loss rates. TTC baseline architecture cannot preserve the FIFO order of arrivals resulting in unfairness among packets. In both baseline architectures, buffer depth grows linearly with the number of baseline stages. Increasing the length of the delay lines does not mean increased buffer depth, as it results in higher packet delays without any improvement in throughput and packet loss rate. For better performances, delay line lengths have to be of one packet length and network operation must be slotted. The TTC baseline architecture with n TTC modules can have the same packet loss probability as the baseline architecture with $2n$ TC modules (Cruz and Tsai 1996). The common feature of these architectures is that entire baseline architecture is controlled by using a distributed control mechanism situated in each module. This feature of the baseline architecture reduces control complexity regardless of the number of stages in the baseline architecture.

In (Cruz and Tsai 1996), it was suggested that fixed length delay lines preserves the FIFO order of arrivals. If the delay line lengths are increased, lower loss performance can be achieved with the cost of disrupting the FIFO order. Moreover, required number of smart switches is reduced. The baseline architecture becomes no longer work-conserving when the input load is relatively light.

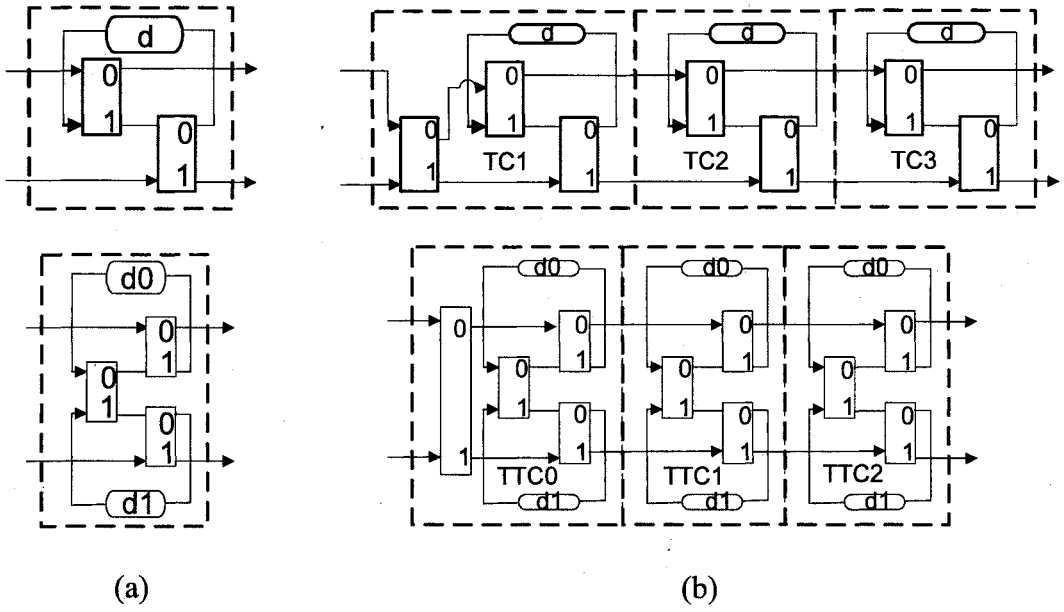


Fig. 2.16 – COD architectures: (a) TC and TTC modules, and (b) baseline architectures for TC and TTC

2.3.3.2. switched fiber delay-line (SDL)

An SDL unit is a combination of 2×2 switches and optical delay lines. It was originally proposed for resolving receiver contentions in a WDM star network (Chlamtac and Fumagalli 1994). It was also used as a node architecture in ring networks, such as FDDI and Quadro double-ring (Queueing arrivals for delayed routing/reception operations) (Chlamtac and Fumagalli 1993a) and multihop optical Manhattan Street Networks (MSN) (Chlamtac and Fumagalli 1993b) together with other versions of this architecture. One possible implementation of SDL is illustrated in Fig. 2.17(a). It consists of d delay lines and $(d + 1)$ binary switches. Normally delay lengths are of one packet length and network operates in a slotted fashion. In (Chlamtac et al. 2000), it was shown that switch performance can be improved by adjusting delay lengths appropriately.

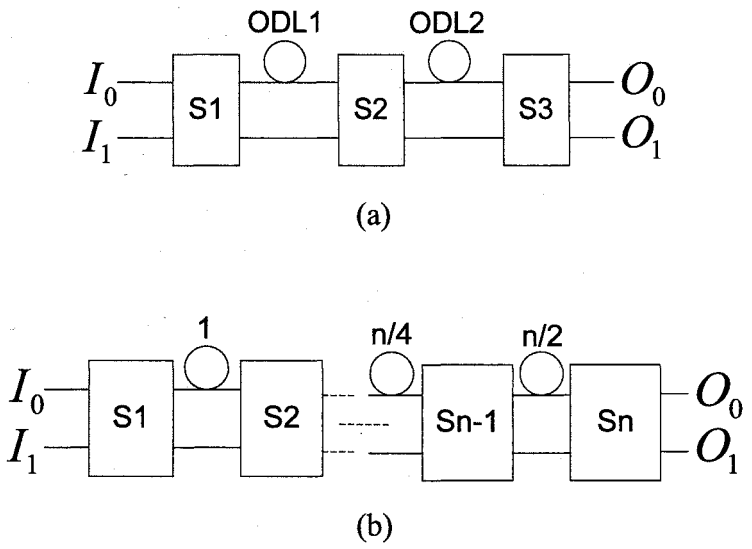


Fig. 2.17 – 2×2 delay-line switch architectures: (a) SDL switch, and (b) logarithmic delay-line switch

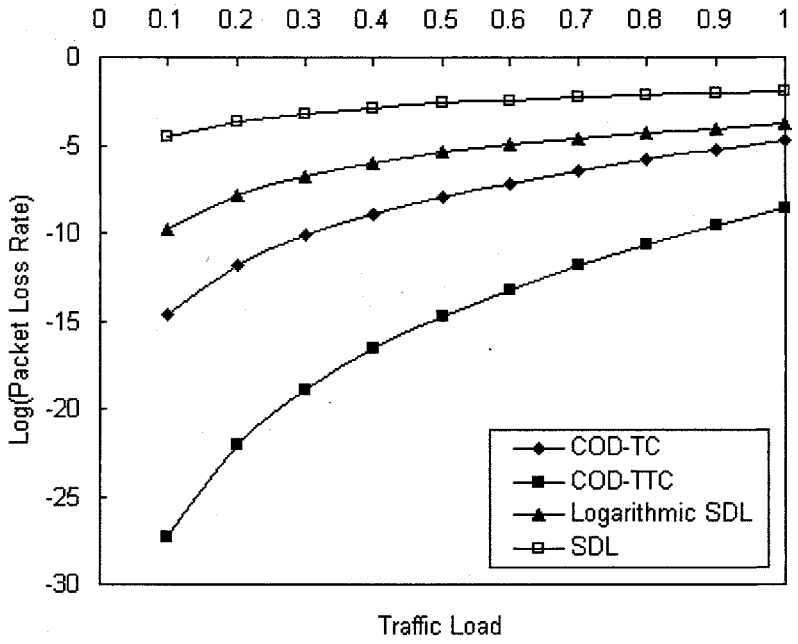
SDL architecture employs a control scheme in order to resolve packet contentions and keep FIFO order of arrivals. The control mechanism is simple and modular. Cascaded architecture can be virtually divided into stages that consist of a switch and one-packet delay. The last switch is used to deliver packets to the correct output ports. Simply, each stage tries to deliver two non-conflicting packets to the next stage of the cascaded architecture, e.g. Type A packets are destined to O_0 and Type B packets are destined O_1. There is no need to keep a track of all packets in the SDL for routing and buffering decisions, as contention resolution decisions are made locally. Packet loss (or deflection) occurs at the outputs of SDL in case when packets exit from the wrong output port of the SDL architecture; that is Type A packet is delivered to O_1 or Type B packet is delivered to O_0.

2.3.3.3. logarithmic delay-line switch

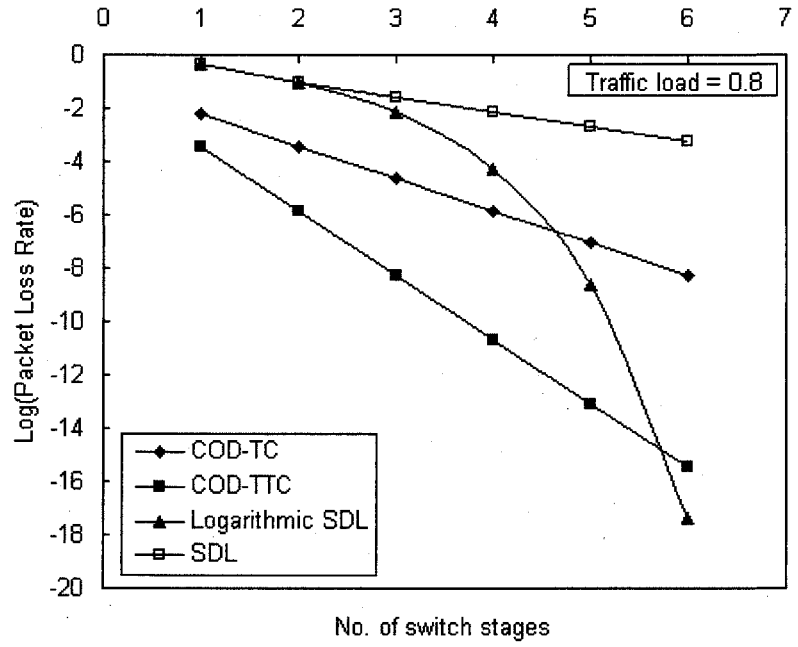
Another version of SDL is the *logarithmic delay-line switch* and shown Fig. 2.17(b). Buffer depth grows exponentially with the number of binary switches in the cascade, $m = 2^{n-1} - 1$, where m and n denote buffer depth and the number of binary switches. Like conventional SDL, it emulates output buffering and functions as a 2×2 output-buffered switch. In heavy traffic load conditions, this architecture offers better loss performance due to unequal delay line lengths in every stage. However, there is a possibility that there exist empty time slots in the delay lines when the input traffic load is moderate. As a consequence, it might not be possible to achieve optimal throughput-delay performance (Hunter et al. 1997). As in SDL architecture, switch control is modular and packet losses occur when packets are delivered to wrong output ports.

2.3.3.4. analysis of 2×2 multi-stage feedforward architectures

Based on the analytical models developed for the COD (Cruz and Tsai 1996), SDL (Chlamtac and Fumagalli 1994) and logarithmic delay-line switch (Hunter and Andonovic 1993), a stand-alone comparison is carried out to justify the packet loss rate performances. Fig. 2.18(a) illustrates the packet loss rate against the traffic load. For this analysis the number of switch stages are selected as four ($k = 4$) for all of the architectures. That is, four-packet capacity for TC and SDL ($m = k$, where m is the buffer size), and eight-packet storage capacity for the TTC ($m = 2k$) and seven-packet storage capacity logarithmic delay-line switch ($m = 2^{k-1} - 1$). The logarithmic delay-line switch performs worse than the TTC due to output idling problem. It is possible to observe similar relation between the TC and SDL.



(a)



(b)

Fig. 2.18 – Packet loss rate versus (a) traffic load, and (b) the number of switch stages for COD-TC, COD-TTC, logarithmic delay-line switch, and SDL

In Fig. 2.18(b), it is shown that for the number of switch stages greater than five ($k >$ 5) the logarithmic delay-line switch outperforms other architectures, as fewer 2×2 basic switches are required in achieving similar packet loss rates as other architectures. For instance, TTC requires nineteen (three 2×2 basic switches per stage) 2×2 basic switches to achieve almost the same packet loss performance as the logarithmic delay-line switch with six 2×2 basic switches. Regardless of the low component cost, the logarithmic delay-line switch suffers from high packet delays and output idling due to lengthy delay lines. On the other hand, the SDL must have at least one packet waiting in ODLs and another packet destined to different output entering the switch in order to be work-conserving. This feature of SDL is known as pipelining.

2.3.3.5. switch with large optical buffers (SLOB)

SLOB cascades many OASIS switches to form a larger switch with a greater buffer depth (Hunter et al. 1998b). OASIS switch is optional and other types of switches can be used as well. SLOB has m inputs and m outputs and delays each packet by between zero and m^i-1 (Fig. 2.19). SLOB emulates an output-buffered switch as it has identical throughput/delay performance. Primitive switching element (PSE) is used to connect two consecutive delay line stages. PSE control logic ensures that no two packets are placed in the same delay line in the same time-slot. Despite the physical and performance motivations, SLOB does not scale well in terms of component cost.

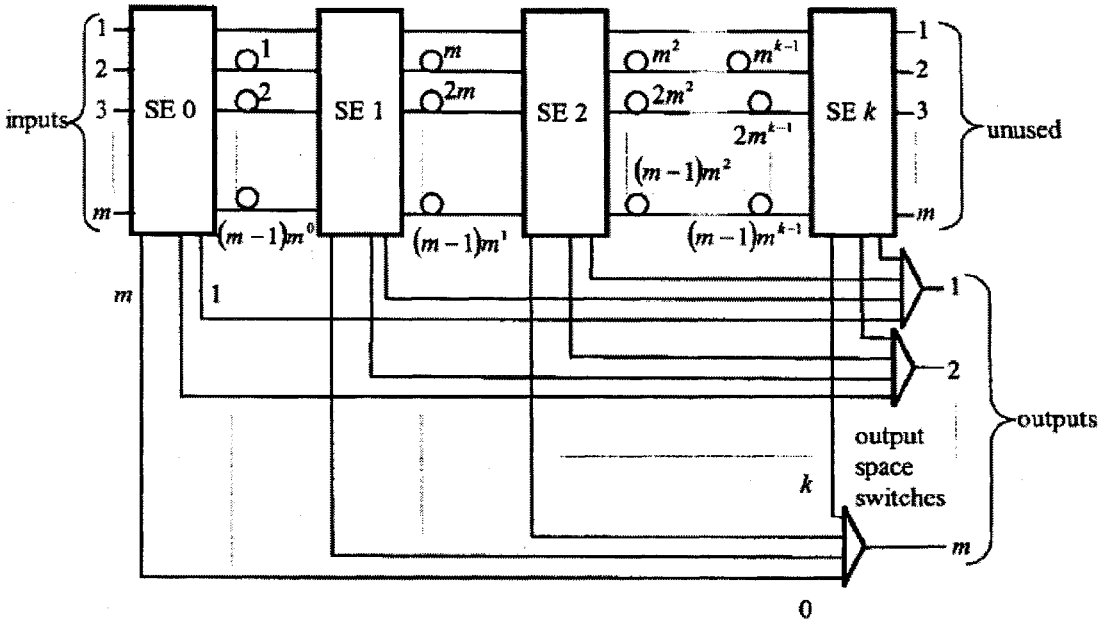


Fig. 2.19 – SLOB (adopted from (Hunter et al. 1998b))

2.3.4. Multi-stage feedback

Multi-stage feedback switch architectures have rarely been considered in the literature. One possible architecture was proposed in (Chlamtac and Fumagalli 1993a) as an 2×2 node architecture for Manhattan street networks. This architecture consists of k 3×3 non-blocking switches and k one time-slot delay lines; (k - 1) delay lines are situated between each pair of switches and the kth delay line is situated on the feedback line (Fig. 2.20). Packet losses occur when delay lines are full and an incoming packet is delivered to a wrong output port. Packets can be recirculated within the architecture for an unlimited amount of time. However, signal amplification is required to avoid degradations on continuous recirculations. Switch control is similar to that of multi-stage feedforward architectures discussed earlier, but switch control must keep a track of recirculating packets to determine whether amplification is required.

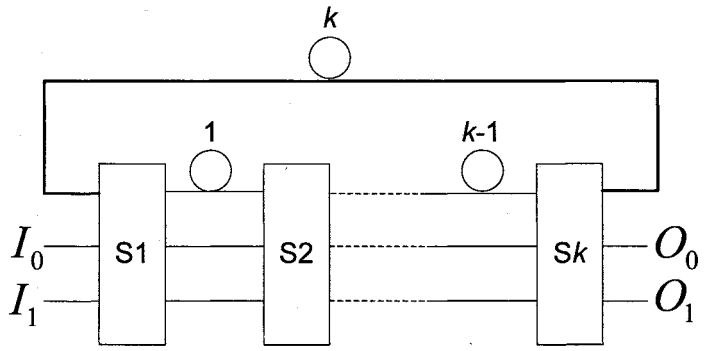


Fig. 2.20 – Optical storage unit (OSU)

2.4. Summary

Due to unscheduled arrival of packets, one or more packets might be destined to a particular output port. Without employing any buffering scheme, only one packet is allowed to leave the switch fabric and rest of the packets are dropped. For this reason packets must be stored as long as contentions persist. In this chapter, an overview of buffering approaches in both electronic and optical domains has been presented. A summary of electronic approaches is provided in Table 2.2. These approaches can be accepted as design patterns for their optical counterparts in terms of performance considerations.

Optical delay-line switch architectures are categorised as single stage feedforward, single stage feedback, multistage feedforward, and multistage feedback. Multistage feedback approaches have rarely been proposed and practical implementations do not exist. Some of the architectures proposed in the literature are summarised in Table 2.3. These approaches are presented in terms of architectural aspects, general performance, type of buffering used, support for priority traffic and control mechanism. Physical shortcomings, such as power penalty, are not considered in this

study. Feedback approaches support priority traffic as packets can be fed back into the switch until contentions are resolved. Feedforward approaches delay packets for a minimum amount of time and employ relatively simpler control mechanisms. Optical delay-line switch architectures require slotted network environment with fixed-length packets to avoid synchronisation and switch control complexities.

Table 2.2 – Summary of buffering approaches in electronic domain

Input queueing	Suffers from HOL line blocking, limiting throughput to 58.6% when $N \rightarrow \infty$. A number of techniques, such as windowing, dropping HOL packets, output expansion, switch speed-up, input port expansion, virtual output queueing, proposed to address this problem.
Virtual output queueing	Proposed to overcome the HOL blocking of input queueing. Modelled as bipartite graph matching problem. Traditional matching algorithms, MWM and MSM, have high complexity. Proposed algorithms aim to reduce the running time complexity of the scheduler and achieve 100% throughput. Widely used in ATM routers.
Output queueing	Switch fabric speed must be N times the line speed of the input channels. Achieves optimal throughput-delay ratio.
Shared buffering	Memory speed must be N times the line speed to admit N packets simultaneously. Achieves similar performance as output queueing with smaller amount of buffers. Achieves optimal throughput-delay ratio. A heavily loaded output port could dominate buffers.
Recirculation buffering	Packets are stored for one time-slot duration. Buffer module is shared among all packets. Packets are fed back to the switch until contentions are resolved. A heavily loaded output port could affect other ports. Switch architecture does not scale well.
Combined input output queueing	Two possible architectures. CIOQ with FIFO and CIOQ with VOQ. CIOQ-FIFO with speed-up factor of two behaves identical to output queueing. CIOQ-VOQ with speed-up reduces scheduler complexity.

Among buffering approaches considered in this chapter, output and shared buffering achieve optimal throughput-delay performance, as packets are delayed only unavoidable amount of time. Shared buffering achieves optimal throughput-delay performance with smaller amount of memory requirement than output queueing due to shared resource allocation. Nevertheless a heavily loaded output port may cause

unfairness among output ports where a heavily loaded output port dominates the usage of shared buffers.

Table 2.3 – Summary of switches with optical buffering

Switch	Architecture	Performance	Type of buffering	Priority	Control
OASIS	Singlestage Feedforward	Same traffic performance as an output-buffered switch.	Shared ODLs. Dynamic length virtual queues for each output port.	NO	Output buffer simulation
Broadcast and select switch	Singlestage Feedforward	Same traffic performance as an output-buffered switch. Provides internal broadcasting.	Each packet encoded with different wavelengths stored in multiple ODLs with different delays.	YES	Output buffer simulation
Staggering switch	Singlestage Feedforward	ODLs are not efficiently shared and utilised. Can accommodate bursty traffic with increased delay line lengths (Karol 1992).	Shared ODLs. Virtual queues for each output port or virtual output queues for each input-output pair.	NO	Output buffer simulation
Multi-wavelength loop switch	Singlestage Feedback	Same performance as an output-buffered switch.	Shared 1-timeslot delay ODL. Packets are stored in different wavelengths.	YES	Shared buffer simulation
SMOP	Singlestage Feedback	Performance close to a shared buffering switch. Switch requires modular design for larger switch sizes (Karol 1993).	Shared multiple length feedback ODLs.	YES	Output buffer simulation
COD	Multistage Feedforward	Packet loss rate decreases when cascading TC/TTC modules. Only 2×2 COD switches exist. TTC architecture cannot preserve FIFO order, but yields less packet loss rate.	1-timeslot delay at each TC module and twin delays at each TTC module. Buffer depth is increased by cascading TC/TTC modules.	NO	Distributed self-routing control mechanism at each module
SDL	Multistage Feedforward	Small scale. Packet loss rate decreases when cascading more delay line stages.	one or more time-slot delay at each stage. Or logarithmic delay line lengths.	NO	Output buffer simulation (FIFO or non-FIFO)
SLOB	Multistage Feedforward	Same performance as an output-buffered switch. Deep buffers help to accommodate bursty traffic.	Different lengths of delay lines at each stage. Delay line lengths increase exponentially	NO	Output buffer simulation

CHAPTER 3

SCALABLE SWITCH ARCHITECTURE WITH DISTRIBUTED BUFFERS

3.1. Introduction

Switching architectures are mainly classified as space division multiplexing (SDM) and time division multiplexing (TDM) (Guizani 1997). Time division architectures, such as bus, ring, utilise a shared medium to transfer packet streams between input-output pairs. In space division architectures, individual paths are established simultaneously between input and output ports. Space division architectures are further classified as single stage, such as crossbar, and multistage, such as Banyan. Other forms of switch architectures in optical domain include wavelength division multiplexing (WDM) (Mukherjee 2000), frequency division multiplexing (FDM) (Guizani 1997), and optical time division multiplexing (OTDM) (Deng et al. 2000).

A general framework for developing a scalable switch architecture with distributed buffers is presented. In this chapter, performances of both electronic and optical node architectures in a blocking-type interconnection network are studied and investigated via simulations. The next section describes the scalable switch architecture that can be built from buffered binary switching elements. Section 3.3 presents suitable buffered node architectures. Finally, simulation model for evaluating scalable switch architecture with various types of buffering techniques is described in Section 3.4 and simulation results are provided in Section 3.5.

3.2. Scalable Switch Architecture

Scalable high-speed packet switches, known as interconnection networks, can be built by interconnecting off-the-shelf switching elements (SEs). In this context, it is possible to define scalability as follows. A scalable switch architecture is the one that offers feasible implementation cost (e.g. switching elements, memory) and control complexity, and desired performance provisions for given switch dimensions.

A typical interconnection network consists of a number of SEs, which are configured in a way to form the desired interconnection pattern between the inputs and outputs. A single-stage interconnect has N input and N output ports with one switching element (e.g. non-blocking space switch). A multistage interconnect has many stages of interconnected SEs based on a network topology. Binary SE consists of two inputs and outputs, and has four possible arrangements: bar, cross, upper broadcast and lower broadcast. Multi-stage interconnection networks (MINs) are essentially classified into three groups (Walrand and Varaiya 1996): blocking, rearrangeably non-blocking, and non-blocking. A MIN is blocking if two or more packets from different input ports cannot be transmitted to corresponding output ports simultaneously. This is mainly due to link conflicts of simultaneous connections between inputs and outputs at different network stages. In rearrangeably non-blocking, non-conflicting connections can be established by rearranging the switch configuration at different stages, where internal blocking occurs (e.g. Benes). Unlike blocking type networks, rearrangeably non-blocking interconnects have more than one path connecting input and output ports. On the other hand, in non-blocking networks, multiple paths can always be established between an input-output pair without any rearrangements.

Blocking-type MINs, also known as single-path MINs, are more favourable due to simple control and low cost as well as performance provisions (Zhou and Atiquzzaman 2002). There are different types of blocking-type MINs, such as Omega, Baseline, Banyan, Flip, depending on the interconnection patterns used (see Fig. 3.1). Despite the difference in interconnection patterns, all exploit the following common features:

1. Single-path MINs are constructed using 2×2 SEs and interconnection pattern (e.g. perfect shuffle, butterfly permutation, cube permutation etc.).
2. In an $N \times N$ MIN, there are $N/2$ rows, $n = \log_2 N$ stages, a total of $N/2.\log_2 N$ SEs and $2N.\log_2 N$ cross-points (A crossbar switch has N^2 cross-points).
3. Simple self-routing mechanism (see Fig. 3.2).

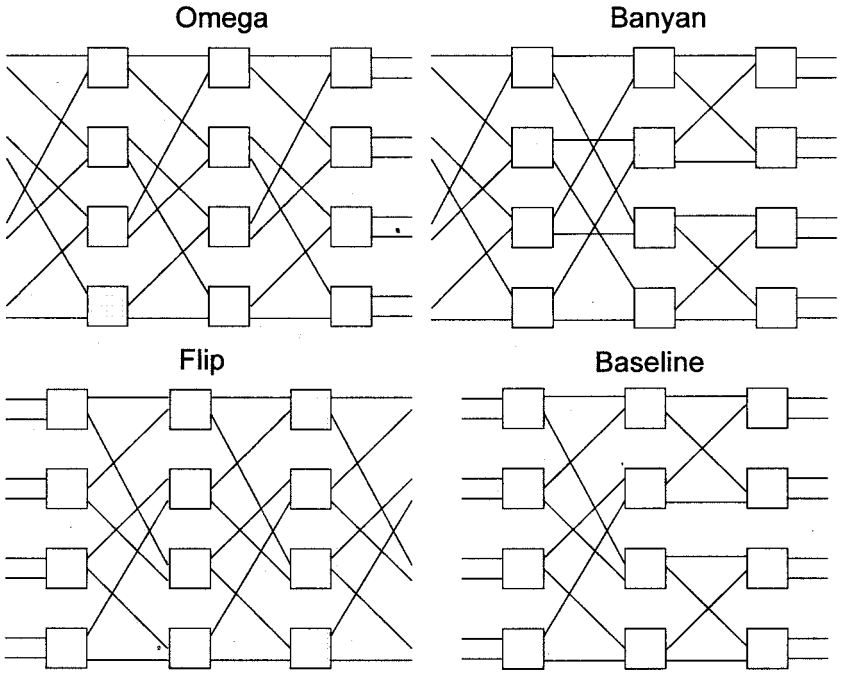


Fig. 3.1 – Interconnection patterns in blocking type MINs

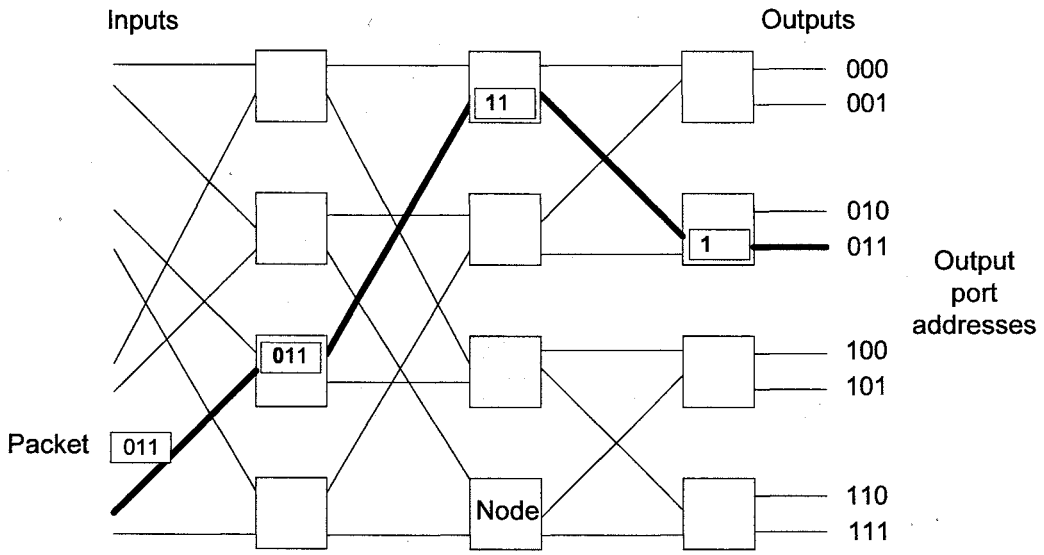


Fig. 3.2 – Self-routing mechanism

When packets arrive at the interconnect inputs, they are assigned with a binary tag. In an interconnect with $n = \log_2 N$ stages, n-bit address tag is attached to each packet (tag = b_0, b_1 ..., b_{n-1}). When a packet arrives at a SE at stage n-1, (n-1)th address bit is checked and the packet is routed. Routing decision can be formally represented by the following:

$$\text{At stage } i \begin{cases} b_i = 0, & \textit{upper port} \\ b_i = 1, & \textit{lower port} \end{cases}$$

This simple routing mechanism and the need for fewer number of SEs make these networks attractive for packet switched systems, such as ATMs in broadband ISDN networks and shared memory multiprocessor systems (interconnecting processors and memories) (Zhou and Atiquzzaman 2002). Nevertheless, lack of diversity within the network architecture results in discarding some of the packets in order to unblock links between inputs and outputs. This yields in high packet loss rates and low throughputs under heavy input traffic. There have been many attempts to tackle this

problem. One popular approach replaces SEs with buffered SEs, called distributed buffering (Zhou and Atiquzzaman 2002). There are two major factors affecting the performance of a buffered MIN. The first factor is the type of buffering used in individual SEs (e.g. input queueing, output queueing, shared buffering etc.) and the second is the control mechanism of the network. The second factor aims to make the buffering area in the individual SEs visible to whole interconnect. That is, effective buffer size seen by each SE can be increased to reduce packet loss rates. Some of these control approaches are pushout (PO), backpressure (BP), restricted backpressure (RBP), and delayed pushout (DPO) (see (Basak et al. 1997; Choudhury and Hahne 1997; Karol et al. 2003) for an overview).

In the literature, only few logical buffering approaches have been considered for buffered MIN architectures (Zhou and Atiquzzaman 2002). All-optical implementations of distributed buffer MINs have not been proposed and performance implications of these architectures are not known yet. In the next section, candidate node architectures (including photonic switches) to be used with MIN architectures are presented. These architectures will then be simulated to test and evaluate the effect of different buffering schemes in the performance of blocking type MINs.

3.3. Node Architectures

A list of candidate node architectures is provided in Table 3.1. Input queueing achieves poor performance due to HOL blocking. Even though the switch size is 2×2, the throughput performance is limited to 75% (Hluchyj and Karol 1988). Therefore, this buffering approach will not be considered. Virtual output queueing scheme was

simulated and tested as node architecture in (Zhou and Atiquzzaman 2002). However, none of the scheduling algorithms were considered in this study. This buffering approach will be revisited and the maximum weight matching algorithms, such as oldest cell first (OCF) and longest queue first (LQF) selection policies, will be considered. Output queueing and shared buffering offer good performances, therefore, can be readily used in MINs. Here both of these approaches will be used as the benchmark in contrast to other approaches. Recirculation buffering is not suitable for buffered MINs, as it has no performance gains over other buffering approaches when the interconnection network node size is 2×2.

Table 3.1 – Candidate buffering schemes for blocking type MINs

Node architecture	**Feature**	**Suitable?**
Input queueing	Throughput is limited to 75% even when node size is 2×2.	No
Virtual output queueing	Achieves better performance with a scheduling algorithm.	Yes
Output queueing	Achieves optimal throughput-delay performance	Yes
Shared buffering	Achieves optimal throughput-delay performance; also buffers are better utilised	Yes
Recirculation buffering	It has no major gains on small switch sizes and not suitable for MINs with 2×2 SEs.	No
Cascaded optical delay line-TC	All-optical 2×2 packet switch. It has never been proposed to be used with blocking type MINs.	Yes
Cascaded optical delay line-TTC	All-optical 2×2 packet switch with dual delay lines. It has never been proposed to be used with blocking type MINs.	Yes
Switched delay line switch (SDL)	All-optical 2×2 packet switch. Initially used in Manhattan networks.	Yes
Logarithmic delay line switch	Lengthy delay lines are not fully utilised when the traffic load is small.	No

Apart from logical buffering techniques, all-optical buffering techniques will be considered in this study. Optical delay-line architectures can be used as node architectures in MINs for the realisation of scalable fast-packet switching all-optical packet switches.

Cascaded optical delay-line track changer (TC) and twin track changer (TTC) are all-optical 2×2 switches. COD architectures emulate output buffering and can be readily used as a node in MINs. Switched delay line switch (SDL) was used in Manhattan networks (Chlamtac and Fumagalli 1993a), but has never been proposed for blocking type MINs. Logarithmic delay line switch consists of uneven and lengthy optical delay lines. As a result, it suffers from high delays and low throughput when the input traffic is moderate. Even though the component cost of this 2×2 switch is considerably low; it will not be considered here due to reasons stated above.

3.4. Simulation Model

Simulation software has been developed to effectively evaluate the performance of different buffering schemes used as node architectures in interconnection network architecture. Different parts of the scalable switch architecture, nodes, network pattern, traffic sources, have been modelled and implemented by using the C programming language.

3.4.1. Interconnection network model

The MIN architecture in Fig. 3.3 will be considered in simulations. In general terms this MIN architecture is called the Banyan network, as Banyan networks cover a variety of network architectures including Delta, Shuffle-exchange, indirect binary *n*-cube, baseline, flip, and the Omega networks. This particular network can be constructed by using butterfly permutation. Butterfly permutation transposes the least significant bit and the *k*th bit of the index and can be iteratively constructed using the following:

$$\beta_{(k)}(x_{n-1}x_{n-2}\ldots x_{k-1}\, x_{k-2}\ldots x_1\, x_0) = (x_{n-1}x_{n-2}\ldots x_0\, x_{k-2}\ldots x_1\, x_{k-1}) \qquad (3.1)$$

The network is synchronous and time-slotted. Each time-slot can hold at most one fixed-length packet (i.e. slot-length equals the packet length). Therefore, one packet or no packets arrive at the input port of the MIN during a time-slot.

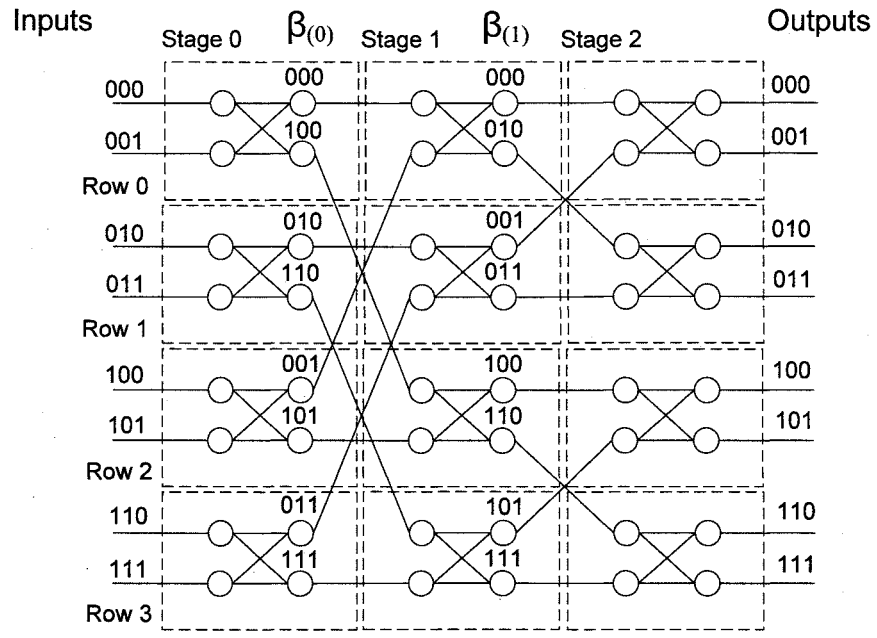


Fig. 3.3 – Banyan network (butterfly permutation)

When a packet arrives at a node, it uses self-routing mechanism of the MIN to reach its destination. The interconnection pattern of the network allows the packet to reach its destination address without having to make any routing decisions. Throughout the simulations it is assumed that there is no propagation delay as packets traverse from inputs to outputs. The following assumptions are made for the interconnection network model:

i. Network is synchronous (i.e. time-slotted)

ii. Fixed-length packets and at most one packet can be allotted per time-slot

iii. No propagation delay

iv. Self-routing mechanism.

3.4.2. Node control mechanism

Each node in the interconnection network employs its own control logic locally and is independent from others. Centralised control schemes, such as backpressure, are not considered in order to reduce control complexity. Besides, all-optical node architectures cannot be used with such control schemes, as packets are delayed rather than stored and also due to architectural restrictions, i.e. feedforward delays, packets cannot be fed back into the node. The control mechanisms intended for node architectures are discussed below.

i. Output queueing and shared buffering:

Control mechanisms for both of these architectures are quite similar. Packets destined for a particular output port of the SE are enqueued at the end of that output port queue. In output queueing, each output port receives an equal portion of buffers. In shared buffering, combined size of the queues must not exceed the total buffer size as a common buffer pool is completely shared among two output ports of the SE. In order to admit two incoming packets to the buffer memory simultaneously, the internal speed of the 2×2 switch fabric must be twice of the line speed.

ii. Virtual output queueing:

In VOQ, for each output port there is a separate fix-length queue at the input ports (see Fig. 3.4). Each incoming packet is first sorted into a queue according to its destination address. Meanwhile each output port selects a packet among queues destined for the same output port according to a selection policy. Due to small node dimensions iterative algorithms have no advantage over relatively complex selection policies. For this reason, (i) oldest cell first (OCF), (ii) longest queue first (LQF), (iii)

random selection, and (iv) hybrid approach (OCF-LQF), selection policies are suitable.

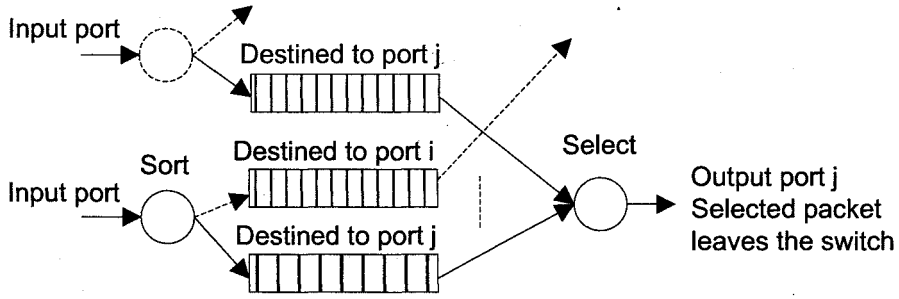


Fig. 3.4 – Sort and select

OCF: Each output port selects an HOL packet with longest waiting time

IF two or more packets have the same amount of waiting time

THEN select an HOL packet randomly

LQF: Each output port selects an HOL packet from the longest queue

IF two or more queues have the same queue lengths

THEN select an HOL packet randomly.

Combined OCF-LQF:

Each output port selects an HOL packet with longest waiting time

IF two or more packets have the same amount of waiting time

THEN each output port selects an HOL packet from the longest queue

IF two or more queues have the same queue lengths

THEN select an HOL packet randomly.

In random selection policy, no information is required about packet waiting times or the queue lengths, because packets are selected in a random manner.

iii. COD-TC and COD-TTC:

The control of both TC and TTC is based on a 2×2 smart switch. Smart switch resolves contentions by using the cross priority scheme. When two packets arrive at the smart switch and destined for the same output port, smart switch becomes cross. One of the packets is delivered to the correct output port whereas the other to the wrong output port. If packets are destined to different output ports, then the smart switch is configured accordingly. Each TC module is composed of two 2×2 smart switches, hence two decision points. Similarly, a TTC module consists of three decision points. Buffer depths can be increased by cascading a smart switch and *n* TC/TTC modules. Each module feeds packets to the next. Despite the cascaded architecture, the control mechanism is still modular. Each module employs its own control mechanism to resolve contentions within a module and tries to deliver packets to correct output ports. If a packet arrives at the wrong output port of the baseline architecture, it is assumed that this packet is lost and it is disposed.

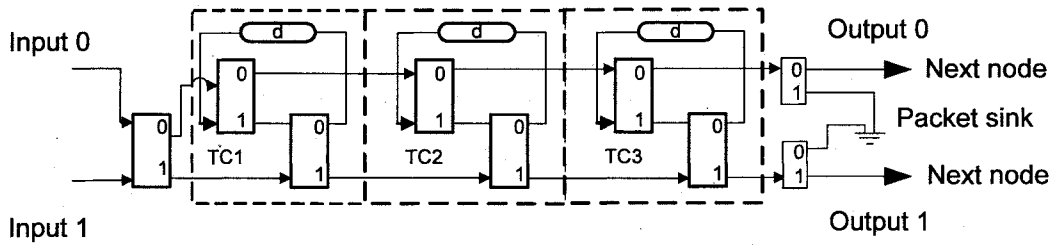


Fig. 3.5 – Modified COD-TC node architecture

Modifications are required in the baseline architectures for realisation of the above claim in an MIN. This modification is illustrated in Fig. 3.5. In this case, another module is added to the cascade to help to remove those packets that arrive at the

wrong output ports. For instance, a packet leaves the baseline architecture from the lower output port, but originally destined to the upper output port, hence has the address bit for this stage of the MIN set to 0 ($b_k = 0$). Therefore, the additional smart switch sends the packet to the correct destination (either packet sink or next node in MIN). In this particular case, packet is sent to the packet sink.

iv. Switched delay line switch:

Implementation of SDL is similar to COD. SDL switch can be formed by using a modular approach. Each module consists of a switch and two output links. One of the links, upper link, has a delay line of one packet length. Cascading these smaller modules form a larger SDL switch with increased buffer depth (Fig. 3.6).

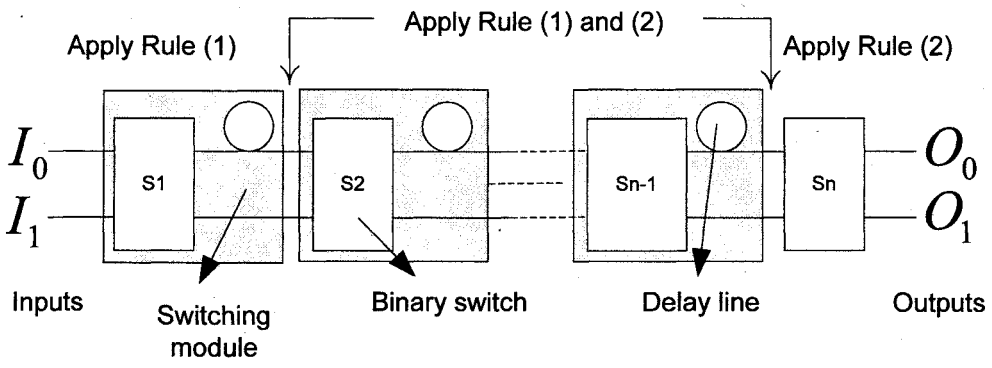


Fig. 3.6 – Switched delay line switch node architecture

Each module uses the same control (except the first and last module) and is managed independently from each other. The control at each module ensures the following rules:

(1) Packets with different output port addresses leave the module. If this is not possible, two packets with the same destination address are delivered to the next module.

(2) A packet emerging from a delay line in the previous module has a higher priority than the packet on the lower link entering the next stage.

A delayed packet is scheduled first and is sent to the link with no delay (the lower link). In order to accomplish Rule (1), the packet can be put in the delay line additionally. In this way the FIFO order among packets with the same destination addresses is preserved. Rule (2) cannot be used for the first module, as both packets arrive from non-delay links. For this reason, if both packets have the same destination addresses, then packets are selected at random. There are n delay lines and $n + 1$ switches in a SDL switch. The two rules outlined above cannot be directly applied to the switch at the last stage, as there are no delays after this stage. For this reason, only Rule (2) is used without having to meet the requirements of Rule (1). This modular control scheme makes overall control of the SDL switch simpler and easy to implement. As in COD, SDL requires an additional module to ensure that packets arriving at wrong output ports are removed from the system.

3.4.3. Traffic source models

Packet arrivals to the input ports of Banyan interconnection network is modelled as an independent and identically distributed (i.i.d.) Bernoulli process. The probability that a packet arrives within a time-slot is constant and the arrivals in each time-slot are independent. Two traffic models used in the simulations are uniform and non-uniform Bernoulli traffic.

Bernoulli arrival process is used to model arrivals in high-speed synchronous systems. Bernoulli is one trial of binomial distribution, $x \sim binomial(1, p)$, and there are two

possible outcomes of Bernoulli process with a given probability of success p. This is defined by the following probability distribution and probability mass distribution functions:

$$F(x) = p^x.(1-p)^{1-x}, \quad for \quad x \in \{0,1\} \tag{3.2}$$

$$P(x:p) = \begin{cases} p, & x = 1 \\ 1-p, & x = 0 \end{cases} \tag{3.3}$$

Where p denotes the probability of a particular event will occur. In the simulations, uniform Bernoulli arrival process is characterised by the following:

- In any time-slot, the probability of arrival of a packet to an input port is p
- Every input has the same arrival probability
- This probability is constant and independent of whether there have been packets on previous time slots and/or other inputs
- Each arriving packet has the same probability of being addresses to one of the MIN output ports and given as $1/N$.

A non-uniform traffic can be formed by using hotspots, where a large proportion of network traffic is directed to a particular output port. The probability of a hotspot output port of receiving packets is more than other ports. Those packets destined to the hotspot port are called the hotspot packets. The probability of each non-hotspot output port receiving packets under hotspot probability H_p is given as:

$$Prob[non-hotspot \quad port] = \frac{1-H_p}{N-1} \tag{3.4}$$

Where N is the number of MIN input/output ports. Equation 3.4 can be generalised for m hotspot ports given as:

$$Prob[non-hotspot \quad port] = \frac{1 - m.H_p}{N - m} \tag{3.5}$$

Where m denotes the number of hotspot ports and $m < N$. During simulations only a single hotspot port case is considered. The aim of using hotspot traffic is to observe the performance implications of non-uniform traffic on different node architectures.

3.4.4. Simulation program

An abstraction of the simulation program is shown in Fig. 3.7 in the form of a flowchart. Simulation program is sequential where each part of the system (i.e. traffic sources and network nodes) is executed one after another. For this reason, the network operates in a time-slotted manner. After all parts of the system are executed in the given cycle (time-slot), the program moves onto the next cycle. This process continues until the maximum number of cycles has been reached.

When the program starts, simulation parameters (i.e. network size, traffic load, hotspot probability, buffer size per node, node type and the maximum number of cycles) are keyed via keyboard. Simulation program then generates the interconnection network (see Section 3.4.1). Until the maximum number of cycles has been reached, the program executes each network node (see Section 3.4.2) and each traffic source (see Section 3.4.3) in every cycle. Each network node performs node specific buffering depending on the switch architecture and buffer management policy used. At the end of each cycle, performance indicators are updated. Finally, at the end of one program execution, results are displayed on the screen and also saved into a text file.

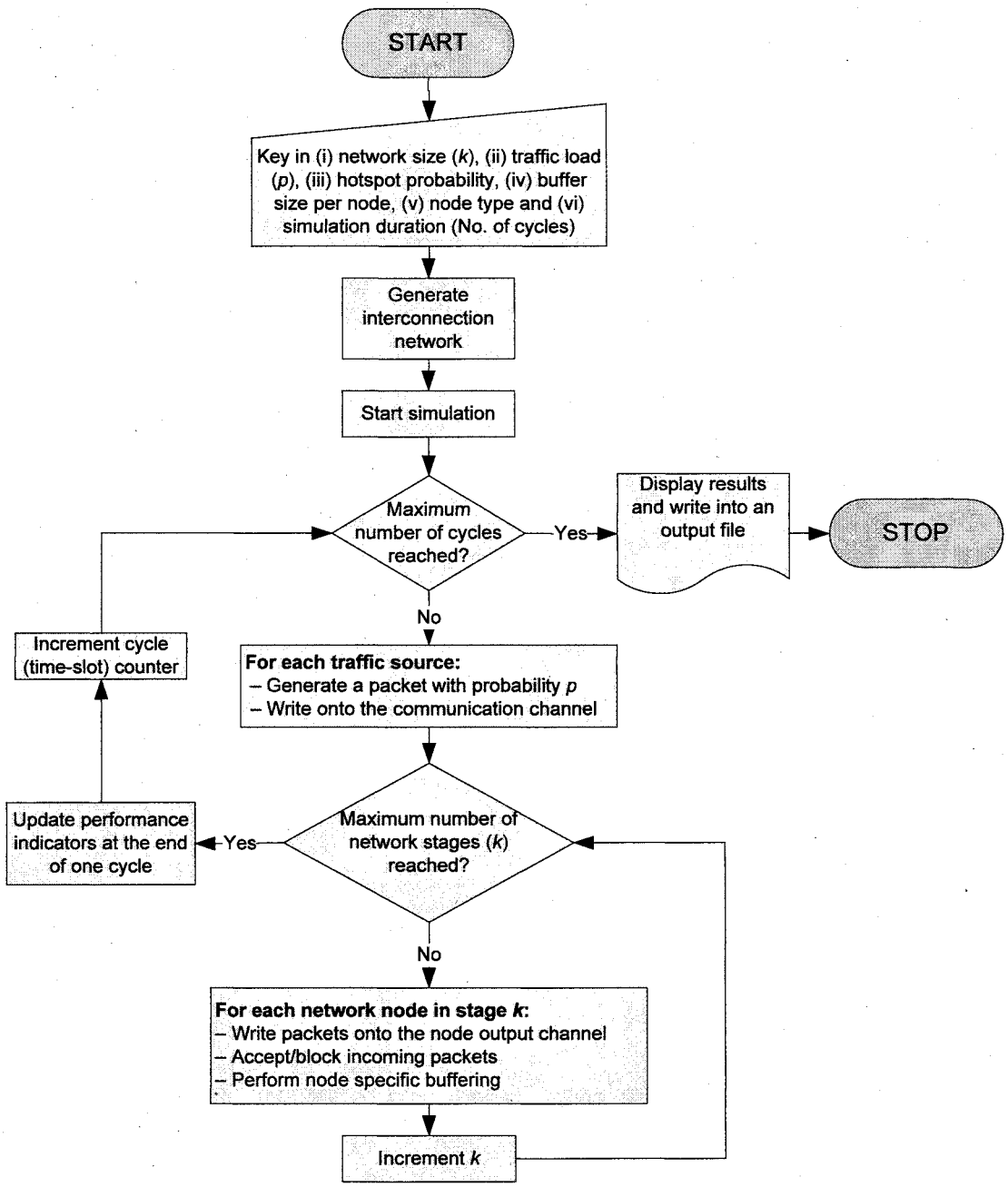


Fig. 3.7 – Simulation program flowchart

3.4.5. Performance metrics

The normalised throughput, average packet delay, packet loss rate, throughput-delay ratio and the average buffer utilisation are used as criteria for comparing the performances of buffering schemes under consideration. Duration of a single

simulation is t_2 = 50,000 cycles (time slots). The values collected until t_1 = 500 cycles are ignored, as the system has not reached steady state. When the network reaches steady state, packets that reach the output ports of the MIN are counted at the end of each cycle. These are averaged over the number of output ports and cycles. Therefore, the normalised throughput τ is given as:

$$\tau = \frac{\sum_{j=t_1}^{t_2-1}\left(\left(\sum_{i=0}^{N-1}\eta_i\right)_j \Big/ N\right)}{t_2 - t_1} \tag{3.6}$$

Where N denotes the number of MIN inputs/outputs, η_i denotes whether a packet is received at the output port i in cycle j, $\eta_i = \{0,1\}$.

The packet loss rate is measured as the ratio of the number of packets lost and the total number of packets injected into the MIN between t_1 and t_2. The average packet delay is calculated by averaging the delay experienced by each packet d throughout the MIN over the number of packets that reach the output ports during $\Delta t = t_2 - t_1$. Each packet header stores the number of time slots delay experienced by the packet. This is incremented only when a packet is waiting in a queue. Traversing from one node to another is not considered as delay and therefore the delay counter is not updated. Only those packets that reach the output ports of the MIN are taken into consideration and dropped ones are not included. Hence, the average packet delay δ is simply given by:

$$\delta = \frac{\sum_{j=t_1}^{t_2-1}\left(\left(\sum_{i=0}^{N-1} d_i\right)\Big/ k_j\right)}{t_2 - t_1} \tag{3.7}$$

Where d_i is the total delay experienced by a packet received at output port i in cycle j, and k_j is number of packets received in cycle j.

Buffer utilisation of the MIN at the end of a cycle is the amount of buffer space occupied in one cycle over the number of available buffer space across the MIN. These are averaged over $t_2 - t_1$ cycles. Average buffer utilisation θ is given by:

$$\theta = \frac{1}{t_2 - t_1}\sum_{j=t_1}^{t_2-1}\left(\overline{Q_j}/Q\right) \tag{3.8}$$

Where Q is the total buffer space in the MIN and $\overline{Q}_j$ is the buffer space allocated in cycle j.

The throughput-delay ratio criterion determines achieving desired throughput while keeping the average packet delay low. Namely, high throughput-delay ratio is an optimal performance for a buffered system, where low delay and high throughput is an important requirement.

3.5. Results and Discussions

The aim of this simulation study is to determine the behaviour of the Banyan network with different buffering approaches under uniform and non-uniform input traffic. A

number of tests have been carried out by varying (i) traffic load (p), (ii) buffer depth, (iii) network size (N), and (iv) hotspot probability (H_p). These metrics will help to assess the performance characteristics of the buffering techniques when used as node architectures in scalable switch architectures, such as the Banyan. The following assumptions are made during simulations:

- The amount of buffer space allocated for each buffering scheme is the same.
- COD-TC and COD-TTC schemes use the same amount of buffering space, hence COD-TC node is $2n$-stage baseline architecture and COD-TTC node is n-stage baseline architecture. Similarly, SDL is a $2n$-stage cascade with the same amount of buffers.
- VOQ uses two times the buffer space that other schemes possess. This is due to dual FIFO queues at the SE inputs.
- VOQ employs the OCF selection policy unless otherwise stated.

3.5.1. Uniform input traffic

(i) Varying traffic load:

This particular simulation study evaluates the effect of input traffic load p. In this setup, the network size is chosen as 64×64, i.e. $\log_2 (2^6)$ MIN stages, and the amount of buffer space (B) allocated for each buffering scheme is denoted as 4-packet storage capacity per node, whereas VOQ node buffer capacity is eight packets. Nevertheless, different network sizes could be considered. The simulation results for the normalised throughput, packet loss rate, average packet delay, and the throughput-delay ratio under varying traffic load are illustrated in Figs. 3.8, 3.9, 3.10, and 3.11. It is possible to analyse the results under four scenarios. Firstly, when buffering is not employed the packet loss rate is substantial (see Fig. 3.9) and therefore the switch throughput is

low (see Fig. 3.8). Note that MIN architecture with no buffers is not included in average packet delay and throughput-delay ratio performance comparisons as packets do not encounter any delay when they propagate through the interconnection network. Secondly, output queueing performs well when compared with no buffering scheme. Packets encounter relatively small delay as they traverse through the MIN (see Fig. 3.10). For this reason, this buffering approach achieves high throughput-delay ratio when compared with other schemes (see Fig. 3.11). Thirdly, VOQ and SDL schemes achieve similar throughput and loss performances. Note that VOQ has double size of the buffers used by the other schemes. SDL switch possesses high delay, as packets undergo unnecessary delay due to architecture limitations. Let's assume that delay lines are empty in a SDL node and two packets destined for different output ports arrive at SDL node. In this case, only one packet is allowed to use the path with no delay lines and the other packet enters the first delay line. It is possible to call this feature pipelining. SDL switch achieves high throughput only when all the delay lines are filled with packets. For this reason, the MIN architecture with SDL switch nodes achieves low throughput-delay performance (see Fig. 3.11). Finally, shared buffering, COD-TC and COD-TTC schemes offer the same performance. Hence, it is possible to claim that COD architectures emulate the shared buffering under uniform traffic. In the shared buffering, buffers are completely shared among output ports and are allocated dynamically in an efficient manner. Hence, shared buffering achieves lower packet loss performance than other schemes. In shared buffering, packets are queued as long as there is sufficient buffer space. Due to high delays, the throughput-delay ratio performance is relatively worse than the output queueing.

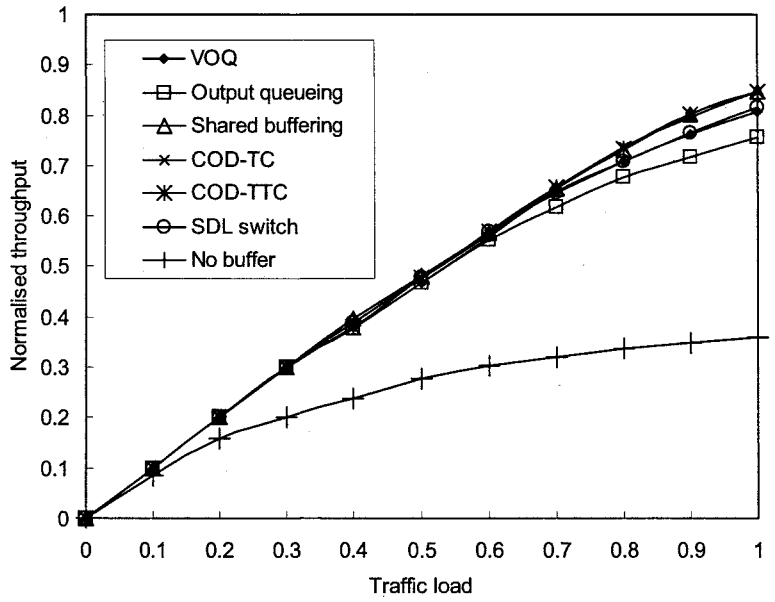


Fig. 3.8 – The normalised throughput versus traffic load under uniform traffic for the MIN architecture with various buffering schemes and without buffers ($N = 64$ and $B = 4$)

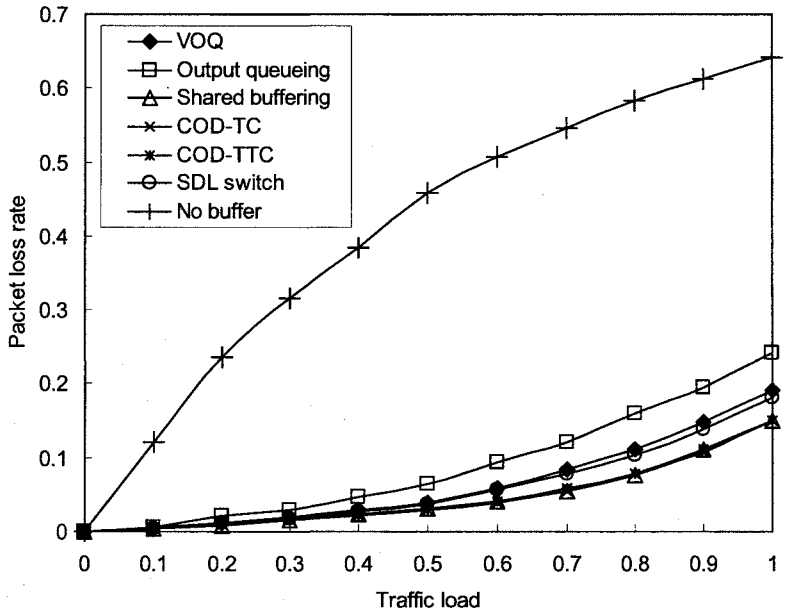


Fig. 3.9 – Packet loss rate versus traffic load under uniform traffic for the MIN architecture with various buffering schemes and without buffers ($N = 64$ and $B = 4$)

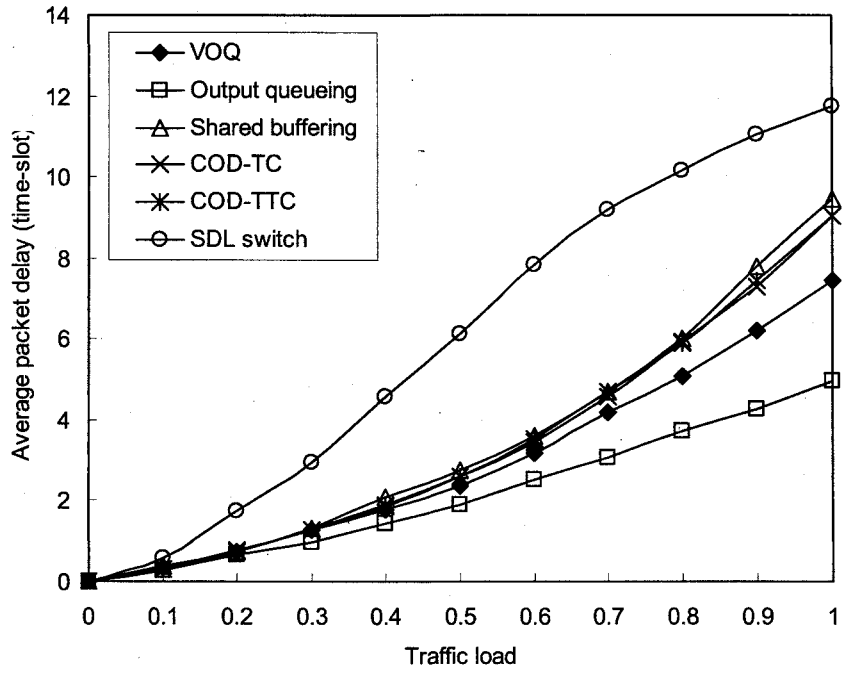


Fig. 3.10 – Average packet delay versus traffic load under uniform traffic for the MIN architecture employing various buffering schemes: VOQ, output queueing, shared buffering, COD-TC, COD-TTC and SDL ($N = 64$ and $B = 4$)

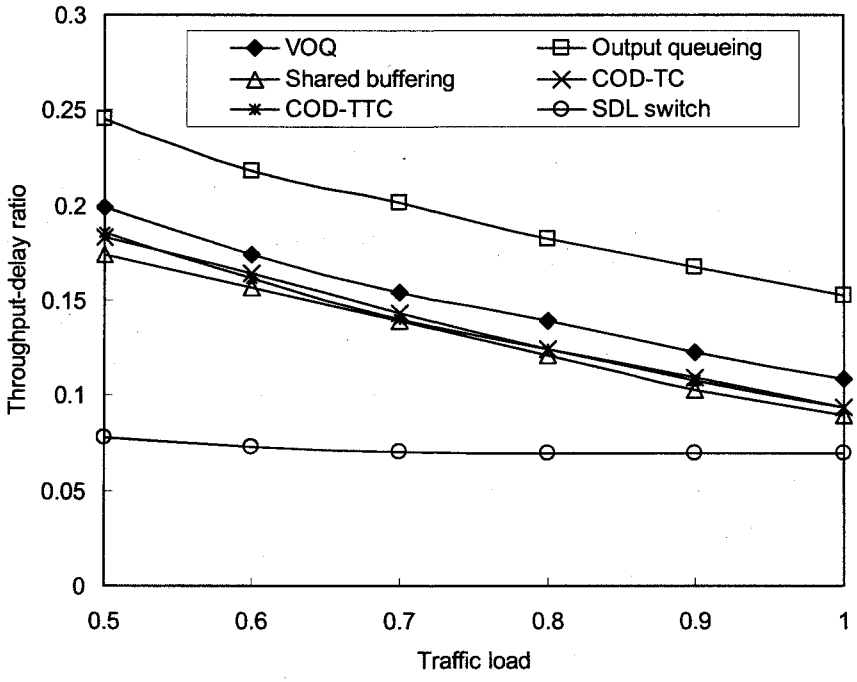


Fig. 3.11 – Throughput-delay ratio versus traffic load under uniform traffic for the MIN architecture employing various buffering schemes: VOQ, output queueing, shared buffering, COD-TC, COD-TTC and SDL ($N = 64$ and $B = 4$)

(ii) Varying buffer depths:

The purpose of this setup is to evaluate the performance implications of buffer depths on the MIN architecture with various buffering schemes.

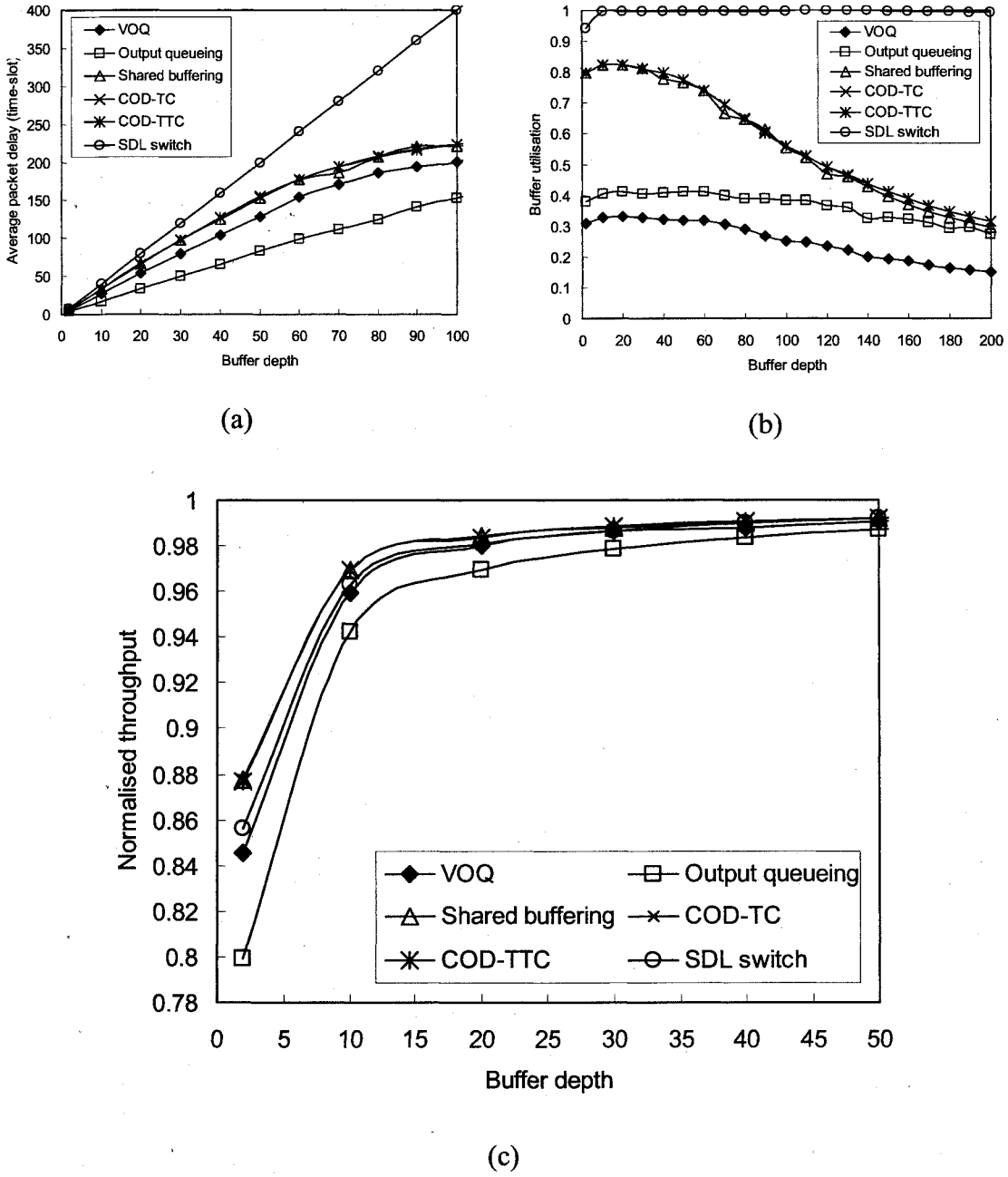


Fig. 3.12 – The effect of varying buffer depths under uniform input traffic for the MIN architecture employing various buffering schemes (p = 1.0 and N = 16): (a) average packet delay, (b) buffer utilisation, and (c) normalised throughput

The impact of buffer depths on average delay, buffer utilisation and normalised throughput are given in Figs. 3.12(a), (b) and (c) respectively. Note that the buffer depth dimension represents the buffer size at each network node (VOQ buffer depth is 2 × Buffer depths). In Fig. 3.12(a), the average delay of the SDL increases linearly with the buffer depth. In Fig. 3.12(b), high buffer utilisation of the SDL is due to the pipelining feature of this architecture but not due to effective use of the buffers. Shared buffering schemes, i.e. COD-TC and COD-TTC, offer the best buffer utilisation. Whilst, the VOQ offers the worst performance due to fixed length dual queues situated at the input ports. Output queueing also offers poor utilisation due to fixed length queues. Regardless of the positions of the buffers and delay lines, the throughput performance of all schemes stabilise when buffers are deep enough to avoid buffer overflow. Disadvantages of using very deep buffers include poor buffer utilisation when the input load is small and extra delay when SDL node architecture is considered.

(iii) Varying network size:

This simulation setup aims to evaluate the effect of varying network sizes on the MIN architecture with different buffered node architectures. The effect of increase in the network size in terms of the number of MIN stages, from $\log(2^3)$ to $\log(2^7)$, is shown in Fig. 3.13. The amount of buffer space (B) allocated for each buffering is four packets per node (whereas VOQ node capacity is eight packets) and the traffic load is one.

The effect of network size on normalised throughput and packet loss rate performances are given in Fig. 3.13(a) and (b) respectively. As the network size

increases, the throughput decreases whereas the packet loss increases. This is because of a much higher than expected packet contentions due to increased number of cross-points. For instance, there are $2\times8\times\log_2(2^3)$ cross-points in an 8×8 switch and $2\times128\times\log_2(2^7)$ cross-points in an 128×128 MIN. As in previous tests in this chapter, shared buffering and COD architectures display similar throughput and packet loss performances for varying network sizes.

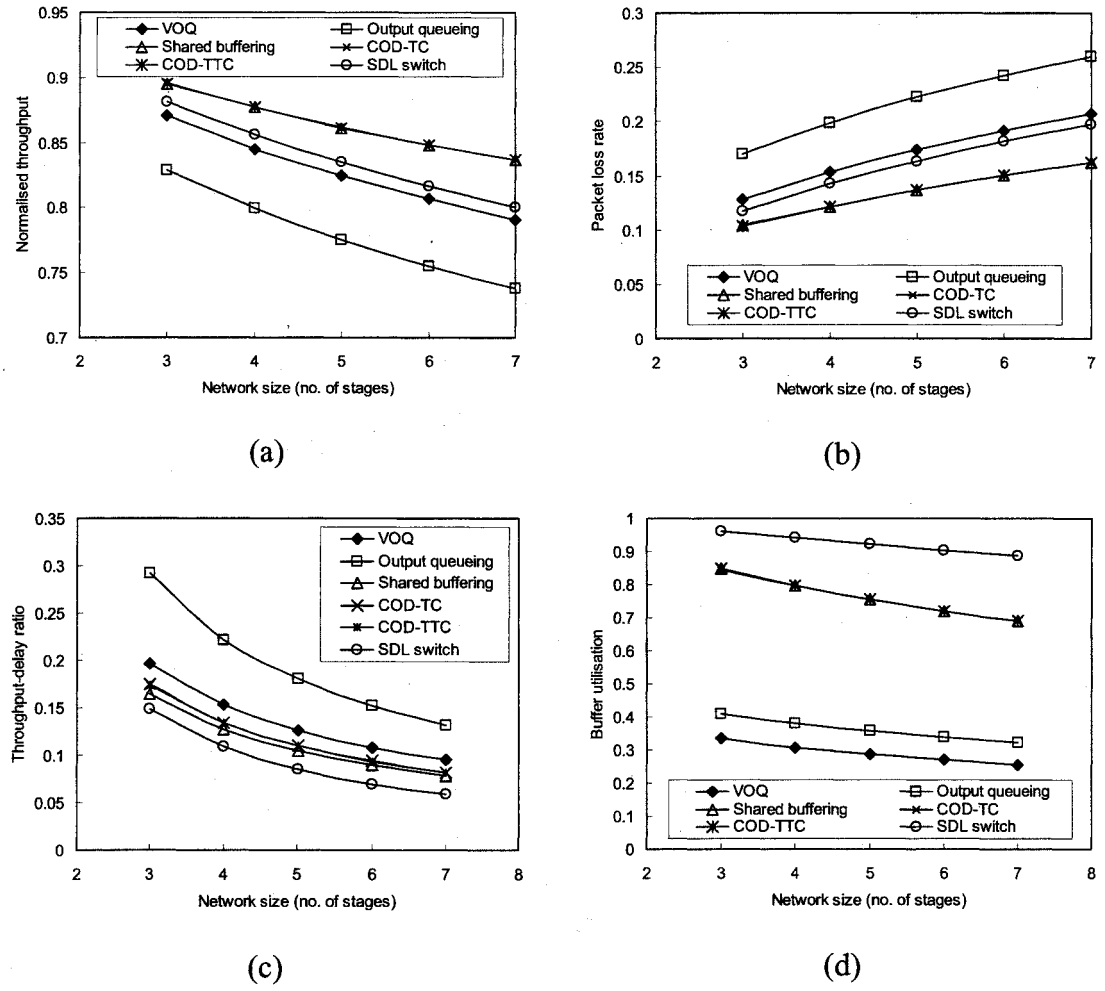


Fig. 3.13 – The effect of varying network size under uniform input traffic for the MIN architecture employing various buffering schemes (p = 1.0 and B = 4): (a) normalised throughput, (b) packet loss rate, (c) throughput-delay ratio, and (d) buffer utilisation

The effect of network size on throughput-delay ratio and buffer utilisation are given in Figs. 3.13(c) and (d). In Fig. 3.13(c), the number of queues that packets traverse increase together with the number of network stages, hence the average delay experienced by each packet. Output queueing offers the best throughput-delay ratio whereas SDL the worst due to its pipelining feature. Shared buffering schemes suffer from high delays as queue lengths are dynamic and queues at each node grow as long as there is sufficient buffer space. In Fig. 3.13(d), overall utilisation for each buffering scheme declines with the network size. Those buffering schemes that employ shared buffering offer better utilisation as sharable buffers increase with the number of node introduced in the MIN. VOQ achieves the worst utilisation for two reasons: (i) twice as the buffer space is used to achieve the same performance as other schemes and (ii) dual buffers used at the inputs of each VOQ node provides inflexible buffer space allocation.

3.5.2. Non-uniform input traffic

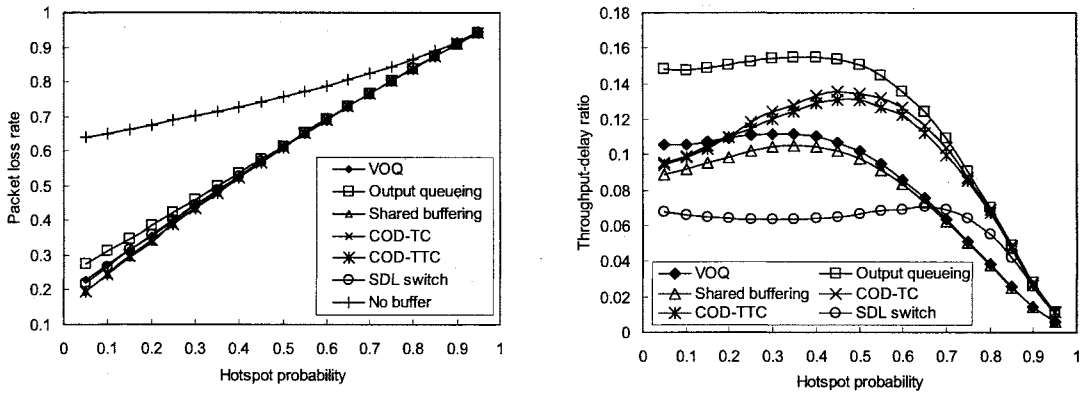
(i) Varying hotspot probability:

The purpose of this test is to determine the effect of non-uniform input traffic. In this case, there is a single hotspot port and the load on this port is changed to observe the influence of the hotspot probability (H_p).

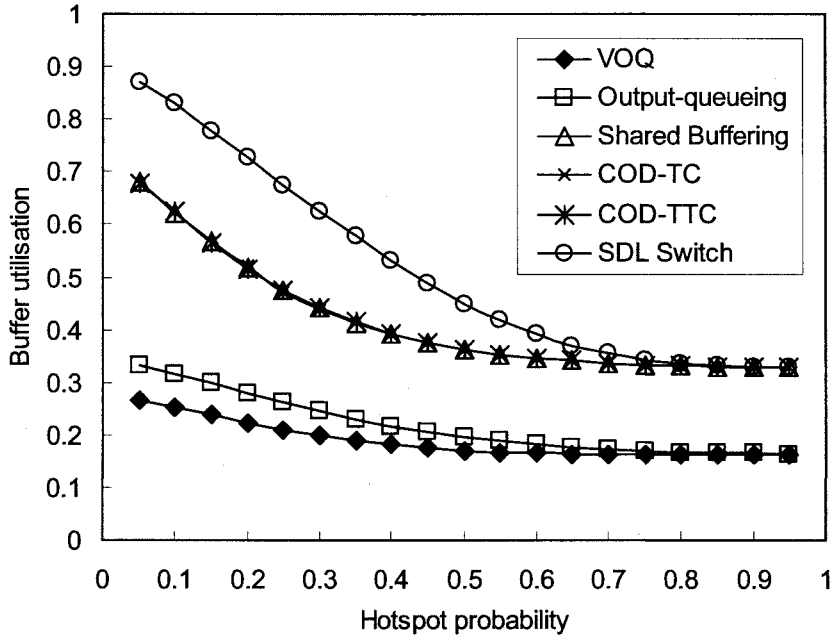
The impact of hotspot probability on packet loss rate performance is shown in Fig. 3.14(a). When the hotspot probability increases, the number of packets destined to a single output port increases. In case of high probabilities, buffers on paths to the hotspot port, i.e. tree-saturation path, become highly saturated whilst the rest of the nodes are left with manageable load. Buffers on saturation path cannot accommodate

all of the hotspot packets thus resulting in high loss rates, known as the tree-saturation effect. All buffering schemes show the same packet loss performance for moderate and high hotspot probabilities. The impact of hotspot probability on throughput-delay ratio is shown in Fig. 3.14(b). In this case, all schemes converge to similar throughput-delay performances as the hotspot probability reaches its peak. This time shared buffering is dissimilar to COD-TC and COD-TTC architectures due to architectural differences (e.g. COD is a modular cascaded architecture rather than a truly shared buffer switch). As in previous tests in this chapter, output queueing achieves better throughput-delay ratio than other schemes from small to moderate hotspot probabilities. An interesting finding is that VOQ matches the throughput-delay performance of shared buffering from moderate to high hotspot probabilities.

Finally, buffer utilisation performances are illustrated in Fig. 3.14(c). Buffer utilisation declines together with high hotspot probabilities in case of all buffering schemes. The reason for this is that network nodes on the tree-saturation paths are underutilised due to most of the traffic being diverted to the hotspot port. Buffering schemes that use partitioned buffers, such as output queueing and VOQ, suffer the most. In a network node, those FIFO queues that are connected to the next stage on the tree-saturation path are better utilised whilst FIFO queues that are connected to other nodes are underutilised resulting in part utilisation of buffers. For very high hotspot probabilities, buffer utilisation of SDL is the same as shared buffering and COD architectures; this is mainly because of the following reasons: (i) SDL behaves like shared buffering when its buffers are fully occupied at heavy traffic loads, and (ii) for very high hotspot probabilities tree-saturation is in effect.



(a) (b)



(c)

Fig. 3.14 – The effect of varying hotspot probability under non-uniform input traffic for the MIN architecture employing various buffering schemes (p = 1.0, B = 4 and N = 64): (a) throughput-delay ratio, (b) packet loss rate, and (c) buffer utilisation

(ii) Varying network size:

The effect of varying network size under imbalanced input traffic with single hotspot port and fixed hotspot probability is shown in Fig. 3.15. Figs. 3.15(a) and (b) illustrate the normalised throughput and packet loss rate performances. As N gets bigger, more

and more packets are directed to the hotspot port. As a consequence, packet loss rate increases together with the network size due to low throughput and inability of the MIN architecture to allocate hotspot packets. Whilst buffers on the tree-saturation path are fully utilised, other nodes are underutilised. All buffering schemes show similar performance patterns as in uniform traffic case apart from increased packet loss rate and low throughput performances.

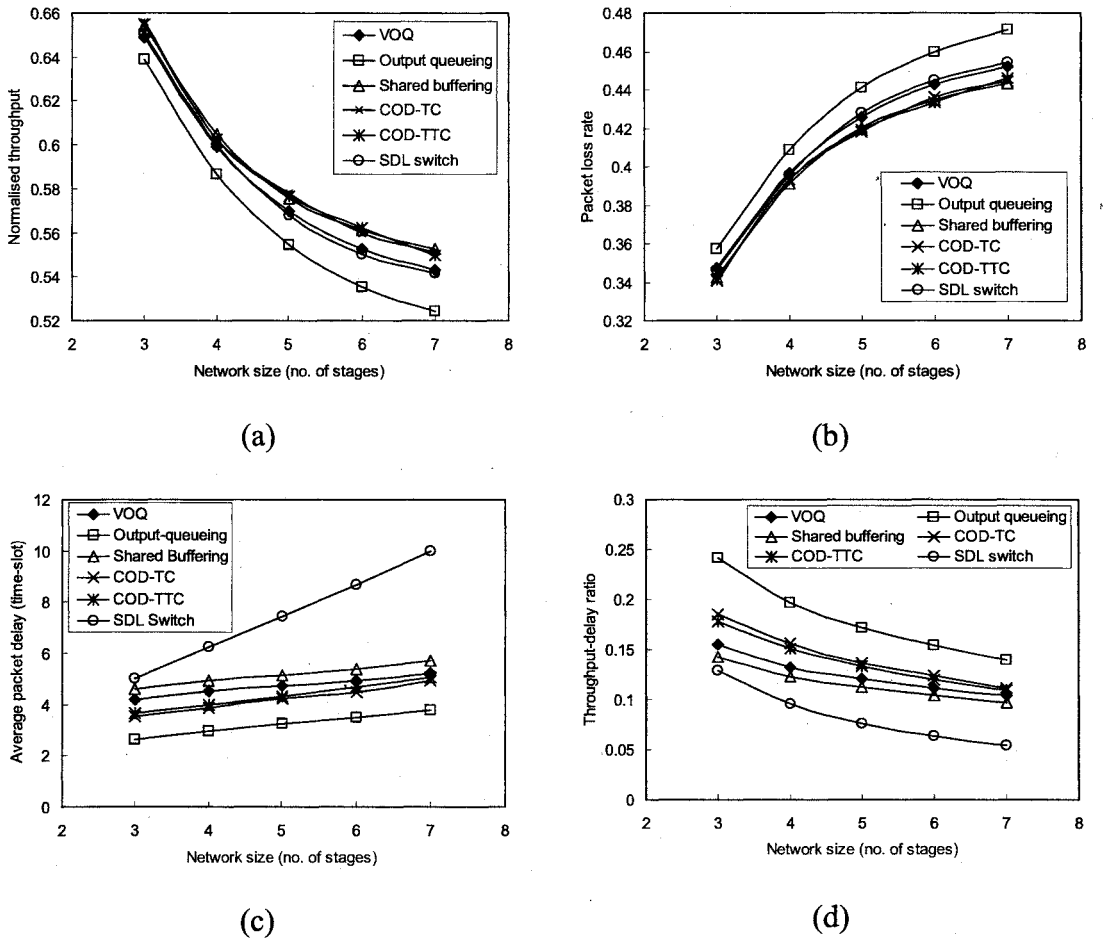


Fig. 3.15 – The effect of varying network size under non-uniform input traffic for the MIN architecture employing various buffering schemes (p = 1.0, B = 4 and H_p = 0.3): (a) normalised throughput, (b) packet loss rate, (c) average packet delay, and (d) throughput-delay ratio

Average packet delay and throughput-delay ratio performances are depicted in Figs. 3.15(c) and (d). All buffering schemes suffer from low throughput-delay performance due to low normalised throughput. SDL switch shows a linear relationship between delay and the number of network stages. Moreover, average delay is more inclined than other schemes due to increased number of network stages that packets pass through as well as pipeline feature of this architecture. Even though output queueing achieves better throughput-delay ratio performance; this is mainly due to low delay experienced by packets. Shared buffering based schemes reveal similar performances, apart from the average delay experienced by shared buffering is higher than COD-TC and COD-TTC.

3.5.3. The 2:1 relation between COD-TC and COD-TTC

In previous tests in this chapter, it has been shown that the Banyan network with TC and TTC architectures achieve the same normalised throughput, packet loss rate, average delay and buffer utilisation performances under both uniform and non-uniform input traffic models when the same amount of buffer space is used. Note that $2n$-stage TC baseline architecture has the same amount of buffer space as n-stage TTC baseline architecture. The aim of this test is to further investigate the packet loss rate performances of the Banyan networks with TC and TTC node architectures to show the 2:1 relation between these node architectures.

Both architectures are tested with different number of modules (from one to ten) and networks sizes (from 8×8 to 64×64). The input traffic load of 1.0 is injected to the network to set the upper bound for the packet loss rate for both architectures. Packet loss rate performances of these architectures are given in Table 3.2. It is observed that

with every possible network dimensions $2n$-stage TC node Banyan network achieves the same packet loss rate performance as the n-stage TTC node Banyan network.

Table 3.2 - The 2:1 relation between TC and TTC for various network dimensions

No. of baseline modules	Packet Loss Rate							
	64×64		32×32		16×16		8×8	
	TC	TTC	TC	TTC	TC	TTC	TC	TTC
1	0.35	**0.24**	0.32	**0.22**	0.29	**0.20**	0.25	**0.17**
2	**0.24**	**0.15**	**0.22**	**0.14**	**0.20**	**0.12**	**0.17**	**0.10**
3	0.19	**0.11**	0.17	**0.10**	0.15	**0.09**	0.13	**0.07**
4	**0.15**	**0.09**	**0.14**	**0.08**	**0.12**	**0.07**	**0.10**	**0.06**
5	0.13	**0.07**	0.12	**0.06**	0.10	**0.06**	0.09	**0.05**
6	**0.11**	0.06	**0.10**	0.05	**0.09**	0.05	**0.07**	0.04
7	0.10	0.05	0.09	0.05	0.08	0.04	0.07	0.04
8	**0.09**	0.05	**0.08**	0.04	**0.07**	0.04	**0.06**	0.03
9	0.08	0.04	0.07	0.04	0.06	0.03	0.05	0.03
10	**0.07**	0.04	**0.06**	0.03	**0.06**	0.03	**0.05**	0.03

The amount of buffering used in the COD baseline architectures can be increased by adding more modules to the baseline architecture. Increasing the delay line lengths improves neither the normalised throughput nor the packet loss rate. Packets encounter extra delay and also FIFO order of packets is disrupted. Note that TTC architecture does not have the FIFO order of arrivals even though the delay line lengths are unity. The effect of optical delay line length on normalised throughput, average packet delay and packet loss rate for 2-stage COD-TC and COD-TTC is illustrated Table 3.3. Studies in (Cruz and Tsai 1996) have shown that adding extra delay between TC/TTC modules improves the packet loss rate performance. That is, a chain with n TC modules with additional delays between each TC modules has the same packet loss probability as the baseline architecture with 2^n-1 TC modules where the delay lengths at each module are of one packet length. However, the delay lengths and the delay line locations must be selected appropriately (Cruz and Tsai 1996).

Table 3.3 - The effect of ODL length on the performance

ODL length (No. of time-slot delay)	2-stage TC chain			2-stage TTC chain		
	Throughput	Packet delay	Packet loss rate	Throughput	Packet delay	Packet loss rate
1	0.65	2.26	0.35	0.76	4.54	0.24
2	0.65	4.52	0.35	0.76	9.07	0.24
3	0.65	6.78	0.35	0.76	13.62	0.24
4	0.65	9.03	0.35	0.76	18.15	0.24
5	0.65	11.29	0.35	0.76	22.68	0.24

3.5.4. Performance implications of VOQ selection policy

Performance of the VOQ node can be improved by employing improved selection policies. Performances of different selection policies, namely OCF, LQF, hybrid (OCF-LQF) and random selection, are compared in this section.

Simulation results for varying buffer depths under uniform input traffic for OCF, LQF, OCF-LQF and random selection policies are illustrated in Fig. 3.16. It is assumed that buffer depth values shown on the buffer depth axis are values given for a single FIFO queue in the VOQ architecture. Note that there four FIFO queues at the inputs of a 2×2 VOQ switch. Therefore, the total buffer size is four times the values shown in buffer depth axis. In this particular setup, the network size is 64×64 and the traffic load is given as $p = 1.0$. The LQF selection policy outperforms other policies in terms of throughput and packet loss rate performances. It suffers from high delays and achieves low throughput-delay ratio performance. Random selection policy is the worst in terms of the packet loss rate and throughput performances. This policy achieves improved throughput-delay performance than any other policy, as the buffer space is not utilised efficiently to admit incoming packets. The OCF and the hybrid selection policies achieve similar performances. This is because hybrid policy

employs OCF first and if the selection ambiguity still persists, it employs the LQF policy. As most of the selection is resolved by using the first selection policy, the usage of LQF policy is very limited.

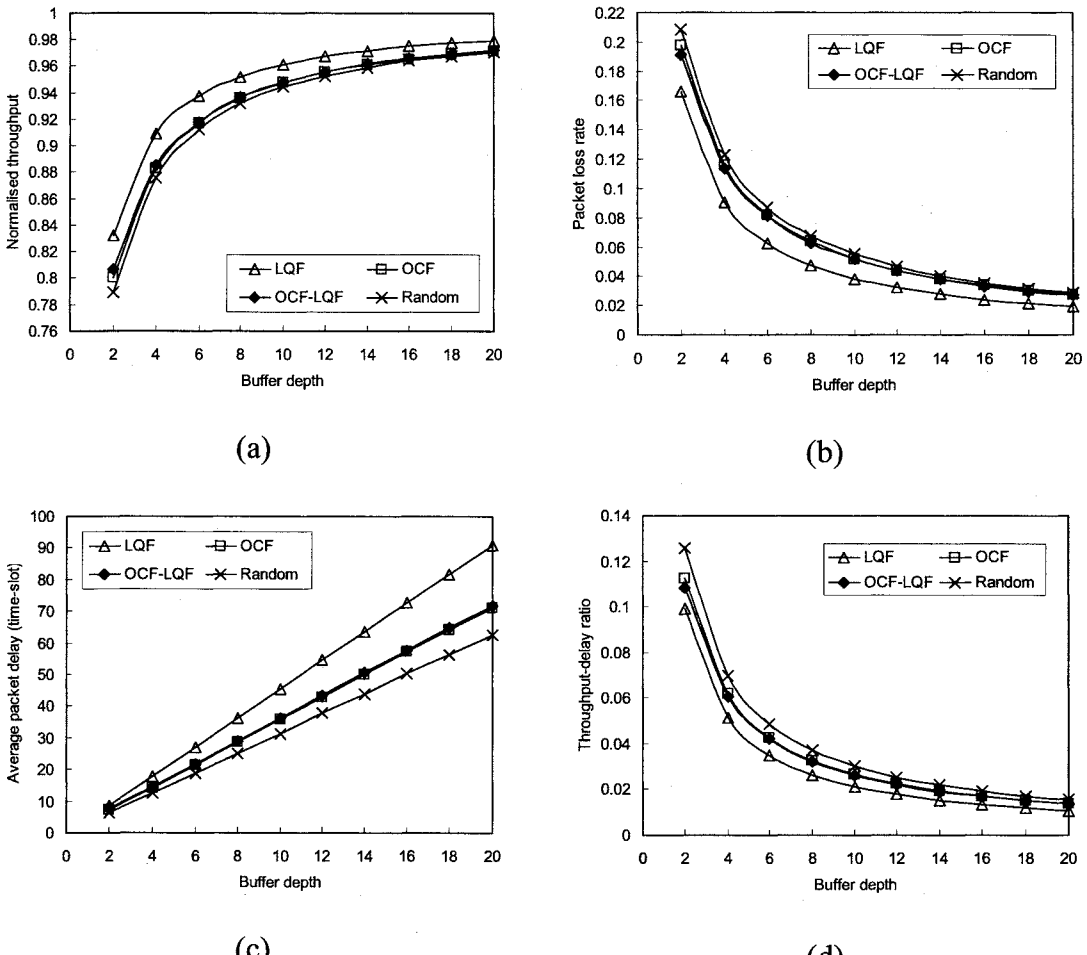


Fig. 3.16 – The effect of varying buffer depth on VOQ with LQF, OCF, OCF-LQF and random selection policies under uniform input traffic (p = 1.0 and N = 64): (a) normalised throughput, (b) packet loss rate, (c) average packet delay, and (d) throughput-delay ratio

In Fig. 3.17, the effect of varying the network size under uniform input traffic is shown. The total buffer size per VOQ node is eight packets (B = 8) and the traffic load is given as p = 1.0. The LQF performs better than other policies for larger

network sizes. For instance, if one wants to achieve very large switch size, LQF is more suitable because of improved throughput and low loss. Note that this policy suffers from high delays. The difference between the OCF and the hybrid approach is obvious in this test. However, it is not possible to draw conclusions on the superiority of one policy over the other. Finally, random selection policy achieves improved throughput-delay performance regardless of the network size.

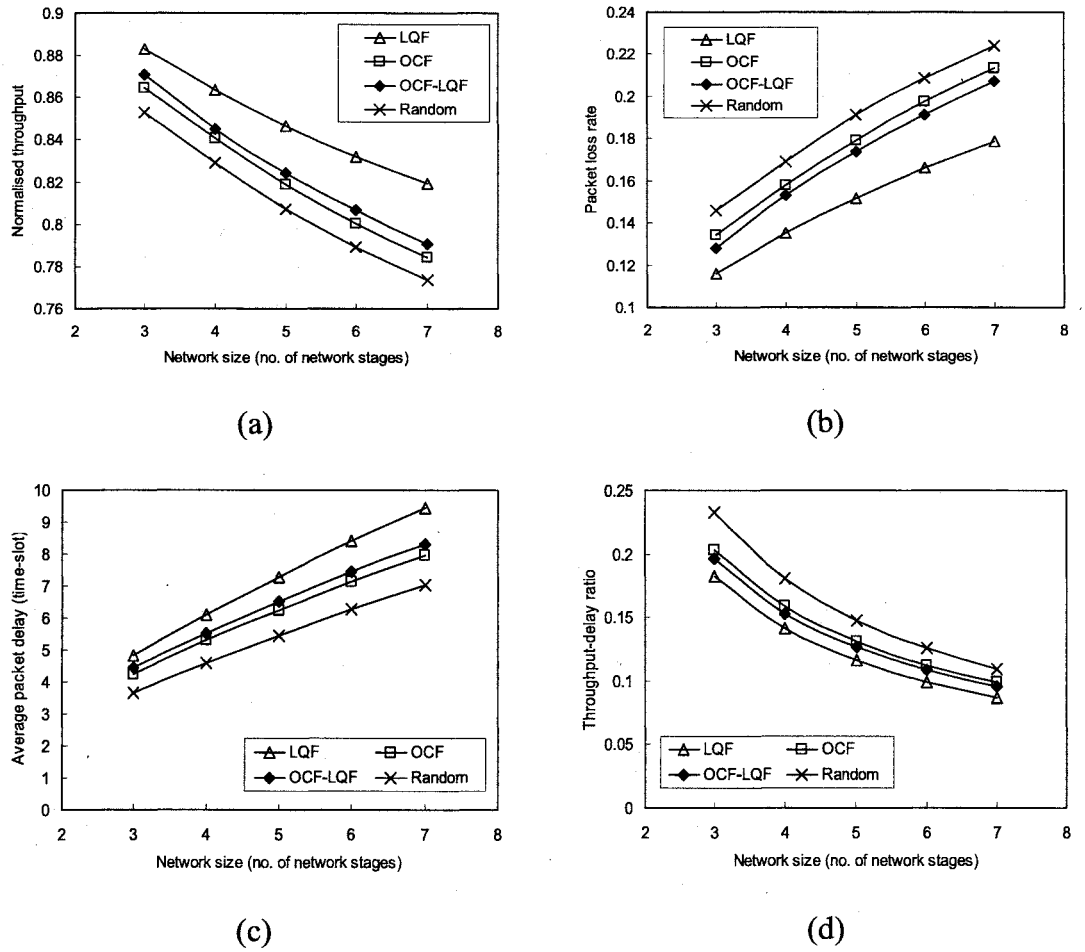


Fig. 3.17 – The effect of varying network size on VOQ with LQF, OCF, OCF-LQF and random selection policies under uniform input traffic (p = 1.0 and B = 8): (a) normalised throughput, (b) packet loss rate, (c) average packet delay, and (d) throughput-delay ratio

The effect of the hotspot load on a single output port is illustrated in Fig. 3.18. In this case, the total buffer size per VOQ node is eight packets (B = 8), network size is 64×64, and the traffic load is p = 1.0. For low hotspot probability, performance differences between selection policies can be spotted easily, whereas all policies display similar performance from moderate to high hotspot probabilities mainly due to the tree-saturation effect.

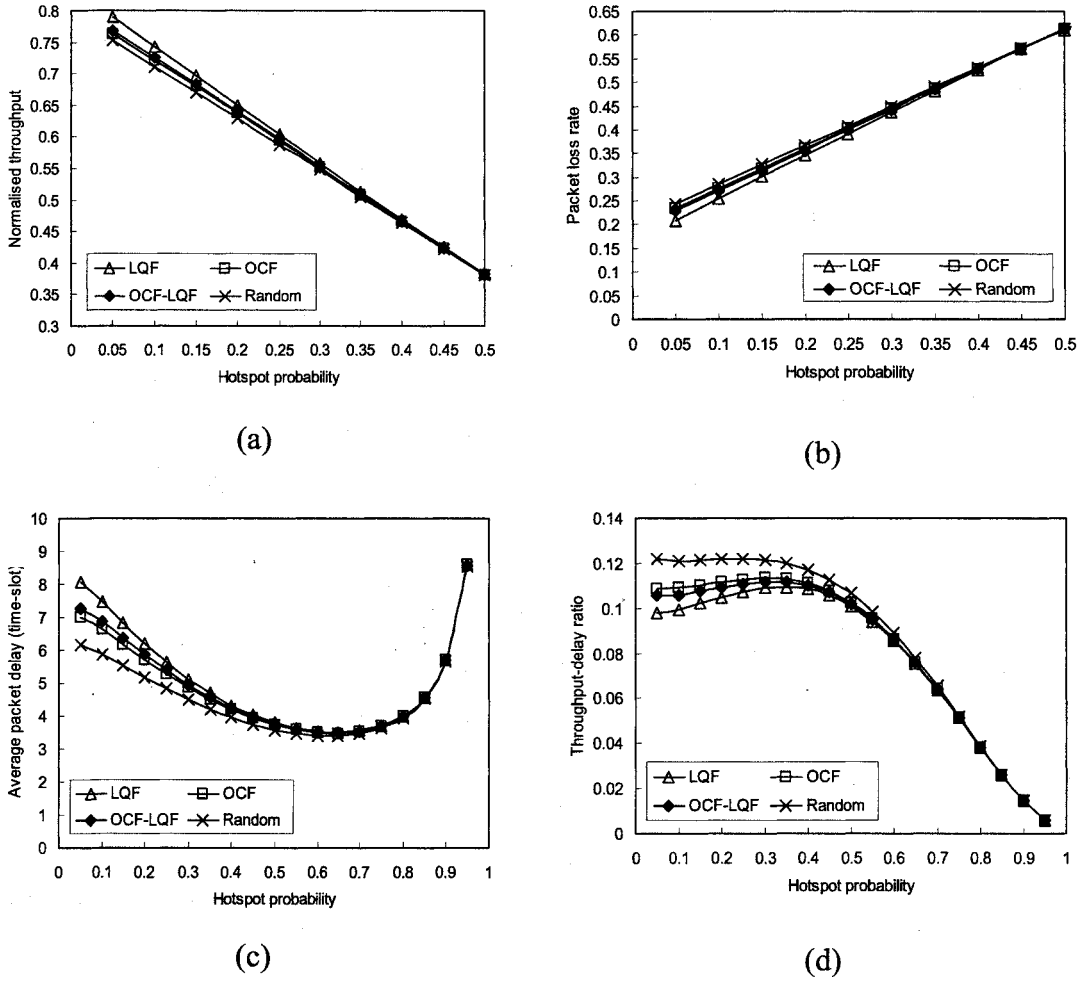


Fig. 3.18 – The effect of varying hotspot probability on VOQ with LQF, OCF, OCF-LQF and random selection policies (p = 1.0 and B = 8): (a) normalised throughput, (b) packet loss rate, (c) average packet delay, and (d) throughput-delay ratio

To conclude, the LQF selection policy offers an improved throughput and packet loss performances over other policies discussed in this section and it is more desirable for network environments where loss performance is important. Random selection policy offers a good throughput-delay ratio performance and it is suitable for the low delay networks. Finally, the OCF and the hybrid policies are somewhat of a trade-off between the LQF and random selection policies. These policies achieve similar performances.

3.6. Summary

In this chapter a general framework for scalable switch architecture with distributed buffers is provided and tested by using different buffering schemes. For the first time, an all-optical MIN architecture that employs optical buffers have been simulated and tested. All-optical 2×2 packet switches are used as node architectures in Banyan like blocking-type interconnection network. In this way very large optical switches can be achieved by using off-the-shelf 2×2 photonic packet switches.

Various tests have been carried out to compare different buffering schemes and to observe the performance of switches with optical buffers in contrast to their electronic counterparts. Note that physical performances, such as signal power etc., are not in the scope of the performance simulations. A summary of results is given in Table 3.4. VOQ and output buffering schemes cannot be scaled for large switch sizes due to partitioned buffers. These buffers cannot be shared among output ports effectively. Whereas shared buffering is suitable for large scale switches, as buffers can be allotted dynamically for output ports. Moreover, buffers in every node can be made

visible across the MIN by employing buffer management schemes, such as backpressure and restricted backpressure etc. VOQ requires selection policies to effectively schedule departure of packets in the next time-slot. VOQ can be tuned by using a suitable selection policy. For instance, the LQF policy provides low loss performance and the random selection policy offers high throughput-delay ratio performance.

Table 3.4 – Comparison of buffering schemes as node architectures in multi-stage interconnection networks

Buffering scheme	**VOQ**	**Output queueing**	**Shared buffering**	**COD-TC**	**COD-TTC**	**SDL switch**
Throughput-delay	Good	Very good	Normal	Normal	Normal	Poor
Delay	Good	Good	Normal	Normal	Normal	Very poor
Packet loss rate	Normal	Normal	Very good	Very good	Very good	Good
Buffer utilisation	Poor	Poor	Good	Good	Good	Very good
Network scalability	Poor	Poor	Normal	Normal	Normal	Normal
Control	Complex	Simple		Simple self-routing, distributed control in every module		Simple control in every SDL stage
Suitable system	Low delay, high-speed, small to medium scale network.		Low packet loss, medium to large scale network.			Low packet loss, delay tolerant, medium to large scale network.

Key: Very poor, poor, normal, good, very good

The COD-TC and COD-TTC node architectures mimic shared buffering and achieve the same performances in most of the traffic conditions. COD-TTC is more economic in terms of component cost when compared with COD-TC. However, it cannot keep FIFO order of arrivals. In both of these architectures, delay line lengths can be increased to achieve improved packet loss performances. This results in high delays

when the traffic load is small and FIFO order of arrivals is disrupted. These architectures are suitable for implementing large scale all-optical packet switches. SDL switch has a distinctive feature among node architectures discussed, known as pipelining. All delay lines must be filled with packets in order to offer reasonable throughput performance. It suffers from high delays in the presence of heavy loads. This architecture might be suitable for delay tolerant networks or multi-processor architectures with optical interconnects. Nonetheless, all three all-optical node architectures require two more switches at the output stages to simply dispose those packets that arrive at the wrong output ports.

During simulations in this chapter, it was shown that imbalanced traffic greatly degrades switch performance resulting in high packet losses and poor buffer utilisation. The rest of this thesis discusses measures required to overcome this problem in shared memory switch architectures.

CHAPTER 4

BUFFER MANAGEMENT IN SHARED MEMORY PACKET SWITCHES

4.1. Introduction

As demonstrated earlier, the shared buffer packet switches are prone to high packet loss rates due to the imbalanced input traffic. Nevertheless, these switch architectures offer good packet loss performances with the requirement of reduced buffer space. In this chapter, the impact of imbalanced input traffic is studied through numerical analysis. Techniques used to overcome performance degradations due to imbalanced input traffic, namely threshold based and pre-emptive, are studied and investigated via simulation of equivalent Markov state diagrams. The next section describes the main features of the shared memory packet switch architecture. A numerical analysis of two extreme policies, complete sharing and complete partitioning policies, used in shared memory switches is presented in Section 4.3. An overview and critical analysis of buffer allocation policies, namely static threshold, dynamic threshold, adaptive control and pre-emptive, are provided in Section 4.4.

4.2. Shared Memory Switch Architecture

Shared memory switch architecture consists of N inputs/outputs, a memory pool shared among output ports and control logic (Fig. 4.1). When packets arrive at the switch inputs, they are multiplexed and read into a shared memory. Write control is responsible from reading packets into the memory locations assigned by the control

logic unit. Read control is responsible from reading the next packets out of the memory according to FIFO order. Control logic is responsible from storing an exact representation of the memory state, updating the state with information from write/read control units, keeping linked-list representations of packets that queue for the same output ports. Queues are represented in a linked-list form, as packets are stored in random address locations in the memory. Often the control unit employs buffer allocation policies for sharing common memory resource in a more efficient and effective manner.

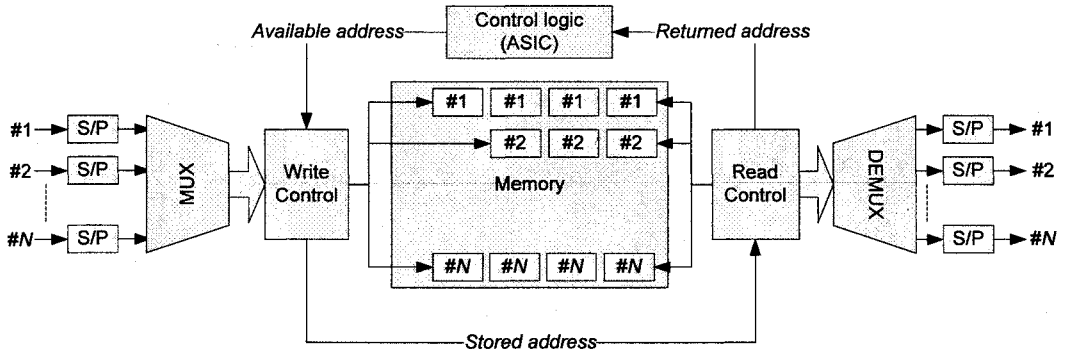


Fig. 4.1 – Shared memory switch fabric

The advantage of shared memory switch over other buffered switches is that its performance is superior to other buffering schemes for the same amount of memory (Guizani and Al-Fuqaha 2001). Even though the performance of shared memory switch is significant, the major disadvantage is the requirement for complex memory management, i.e. memory write speed must be *N* times the line speed. This is because *N* different streams are combined into a single stream in order to be read into a shared medium. High rate packet-interleaving time division multiplexing (TDM) bus can be used to speed up the memory write process, as well as handle packet arbitration process (Engbersen 2003). Spontaneous arrivals of two or more packets that are

destined to the same output port require sequencing and arbitration. Packet arbitrator employs a randomisation process where it selects the next packet for multiplexing and therefore packets are admitted to the corresponding FIFO queue in a consecutive order (Kozaki et al. 1991).

An abstract version of shared memory switch is illustrated in Fig. 4.2. As can be seen from the figure, buffer space is completely shared among N output ports. Queue lengths are subject to change, i.e. dynamic length. Nevertheless, the sum of lengths of all queues must not exceed the pre-defined buffer size and given as:

$$M \geq \sum_{i=1}^{N} q_i(t), \tag{4.1}$$

Where M denotes the buffer size and q_i is the length of queue i. For fixed-size packet networks, such as ATM, queue lengths are measured as number of fixed-length packets (cells) in buffer. In case of variable-length packets (i.e. IP), memory occupancy can be measured as number of bytes in memory. In fixed-size case, buffer is either full or have available space for k packets. However, in variable-size case, even though there are m bytes of free memory, a packet with size m_p bytes cannot be admitted to memory, if $m_p > m$. For this reason, buffer utilisation becomes the main concern and is highly dependent on the choice of optimal packet size range. One can easily state that bigger the packet size longer it takes to transmit packet onto the communication channel. Hence, packet memory cannot be cleared immediately and newly arrived packets cannot be admitted to switch buffers. Apart from this, the choice of optimal packet length is also dependent on packet header overhead and packet error probability factors (Schaar et al. 2003; Yin and Agrawal 2004).

Increasing the packet size reduces the header overhead, which also increases the packet error probability. It is clear that there is a trade-off between packet header overhead in small packet fragments and probability of error in lengthy fragments.

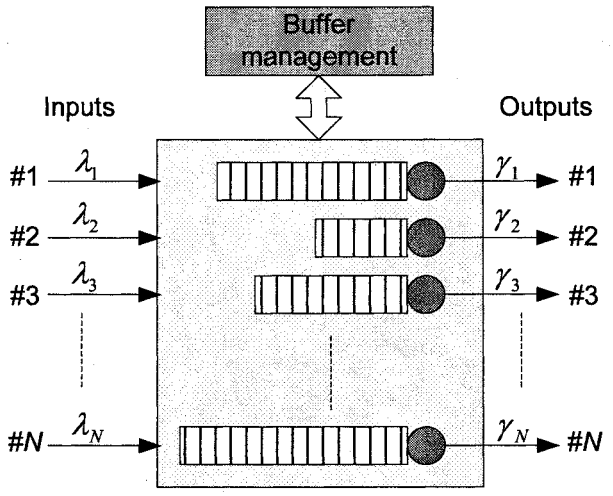


Fig. 4.2 – Shared buffer switch logical structure

One of the main drawbacks of this type of buffering is that heavily loaded output port could affect other output ports (Wei et al. 1991; Arpaci and Copeland 2000). That is, if one of the output ports is receiving a volume of traffic higher than others, then it will require more buffer space; therefore it dominates the usage of buffers.

4.3. Complete Partitioning Versus Complete Sharing

In this section, two extreme situations that are used in the control of shared memory switch are considered. These are complete partitioning (CP) and complete sharing (CS). In CP, buffer space, M, is equally partitioned among N output ports, M/N:

$$M = \sum_{i=1}^{N} k_i, \quad \begin{matrix} k_1 = k_2 = \ldots = k_N \\ and \quad k_i \geq q_i(t) \end{matrix} \tag{4.2}$$

Where k_i denotes the equal portion acquired by output i, M is the buffer size, N is the number of output ports, and $q_i(t)$ denotes the queue length of queue i at time t. On the other hand, CS distributes buffer space without applying any restrictions on the queue lengths:

$$M \geq Q(t) = \sum_{i=1}^{N} q_i(t) \tag{4.3}$$

Some queues may use less buffer space and others may acquire more buffer space than the fair share size (M/N). Therefore, the shared buffer pool is distributed according to the demands from output ports. In various studies (Hluchyj and Karol 1988; Sharma and Viniotis 1999; Arpaci and Copeland 2000; Guizani and Al-Fuqaha 2001), it is shown that both CP and CS achieve optimal throughput-delay performance, as packets are delayed by an unavoidable amount of duration when two or more packets arrive at the same output port. CS achieves improved packet loss performance than CP, because of its flexible memory allocation capability, and hence improved utilisation of the buffer space. Nevertheless, these conclusions are drawn based on the fact that the packets arriving at the switch inputs have equal chance of being destined to one of the outputs, i.e. uniformly distributed input traffic with non-bursty packet arrival.

In an imbalanced input traffic case, one or more output ports receive more packets than others. This type of port is called the hotspot or heavily loaded port. When a packet arrives at one of the switch inputs, its probability of being destined to one of the hotspot ports is higher than the non-hotspot ports. This type of traffic is called hotspot traffic. In order to analyse the performance implications of CP and CS, the

analysis developed in (Wei et al. 1991) worth revisiting. The following assumptions are made for this analysis:

i. Arrival of packets is modelled as i.i.d. Bernoulli process. Time is divided into equal portions, called slots, and at most one packet is generated in a slot.
ii. Probability of a packet arriving in a time slot is p, so is the total traffic load.
iii. Each packet is destined to a single hotspot port with a probability of r / N, where r is the imbalance factor ($1 \leq r \leq N$). The rest of the ports receive packets with a probability $(N - r)/N$. Each of $N - 1$ output ports receive packets with a probability $(N-r) / [N.(N-1)]$.
iv. Arrivals to each input port and arrivals in successive time slots are independent from each other.

In an ideal case, the sum of all arrivals to shared buffer switch ($\sum_i \lambda_i$) must be the same as departures from the switch ($\sum_i \gamma_i$) at any given time. Arrivals and departures are in equilibrium at all times:

$$\lambda_1 + \lambda_2 + \ldots + \lambda_N = \gamma_1 + \gamma_2 + \ldots + \gamma_N \tag{4.4}$$

When $\left(\sum_i \lambda_i\right) > \left(\sum_i \gamma_i\right)$, fewer packets leave the switch outputs than the arrivals. This may occur when two or more packets are destined to a particular output port (i.e. hotspot port). Even though the switch buffers are infinite and can accommodate contending packets; queue length of the hotspot port grows infinitely. Consequently, the hotspot port can only deliver one packet in a time slot and receive at least two packets. Since switch buffers are finite, some of the packets are rejected from entering the switch buffers.

Now let's consider the CS case. Assume that there is only one hotspot port that receives packet traffic with rate $p.r$. When $p.r > 1$, the arrival rate is greater than the departure rate and the length of this queue grows substantially. Gradually, packets destined to other ports leave the switch and the shared buffer is filled with packets only destined to the hotspot port, i.e. hotspot packets. Finally, buffers become saturated and packets arriving at the hotspot port as well as other ports are dropped. The number of packet lost on hotspot port (e.g. output port 1) is given as, $P_{loss}(i=1) = p.r - 1$. It is acceptable to assume that in steady state the average packet loss probability of each output port is approximately the same. Therefore, the overall packet loss performance of the shared buffer switch with CS scheme is given as:

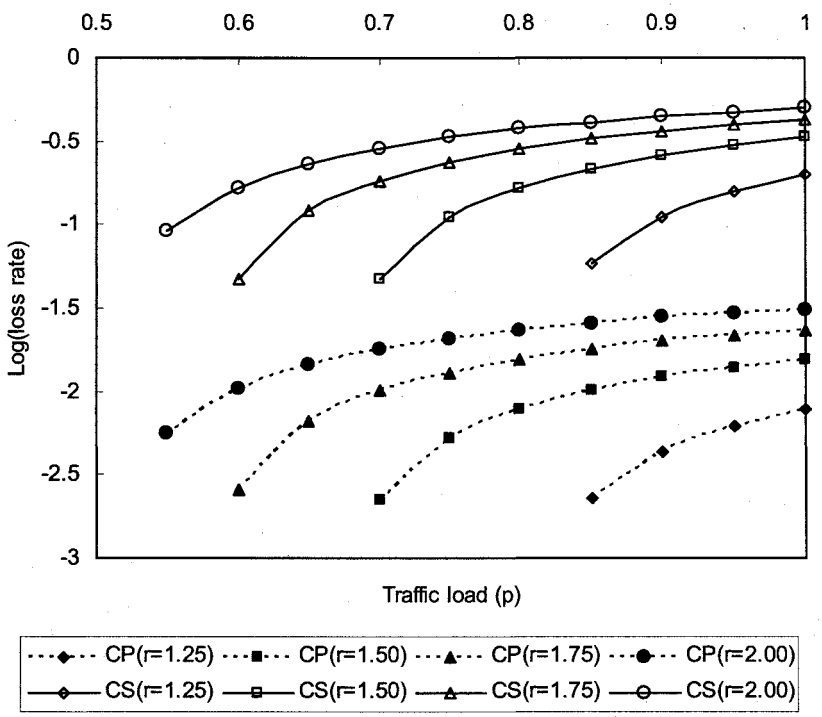
$$P_{loss}(i) = \frac{p.r - 1}{p.r} \qquad \text{for } p.r > 1. \tag{4.5}$$

In case of CP, each output has its dedicated equal portion of the buffer. Imbalanced traffic is in effect on hotspot ports only and the rest of the ports remain unaffected. That is, the packet losses are from those heavily loaded ports, as they are unable to accommodate excess traffic. Hence, the overall packet loss is given as:

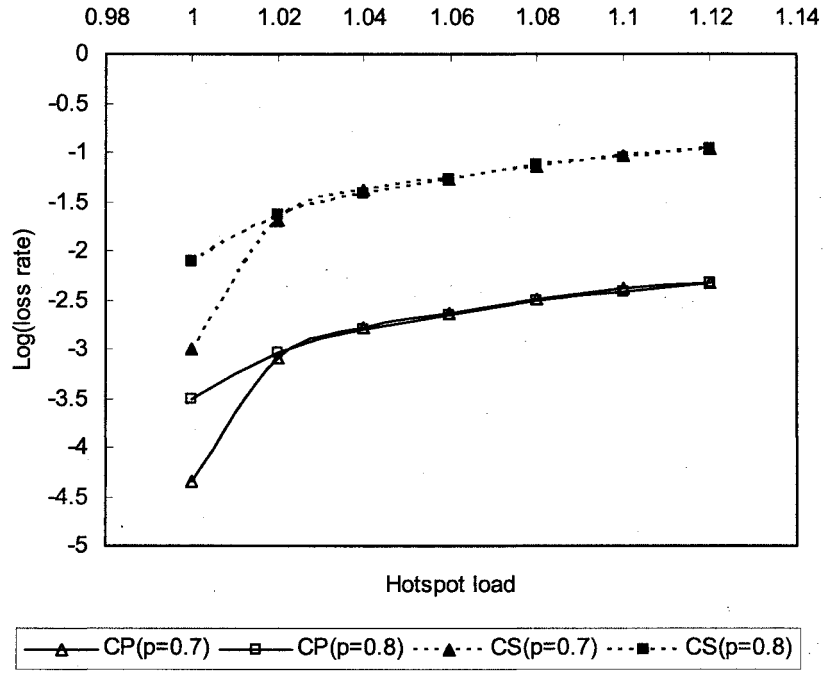
$$P_{loss}(i) = \frac{p.r - 1}{p.N} \qquad \text{for } p.r > 1. \tag{4.6}$$

The above condition is valid for single hotspot port case and can simply be rewritten to include the effect of the number of hotspots as follows:

$$P_{loss}(i) = \frac{k(p.r - 1)}{p.N} \qquad \text{for } p.r > 1 \tag{4.7}$$



(a)



(b)

Fig. 4.3 – The effect of traffic imbalance on CS and CP. Packet loss rate versus (a) traffic load with degree of imbalance, and (b) hotspot load

Packet loss performances of both complete sharing and complete partitioning schemes are illustrated in Fig. 4.3. These results are produced by using Equations 4.5 and 4.6 for various traffic loads p with degree of imbalance r = {1.25, 1.50, 1.75, 2.00}, and various loads on hotspot port $p.r$ with traffic load p = {0.7, 0.8}. In Fig. 4.3(a), complete partitioning can prevent hotspot ports from dominating the buffers due to dedicated allocation for each port and hence achieves better loss performance regardless of the traffic load imposed. In Fig. 4.3(b), the effect of hotspot port load is illustrated. When hotspot load is greater than the service rate, queue lengths increase dramatically. Therefore, regardless of system load, buffer space is occupied by hotspot port packets. When hotspot load is slightly more than the service rate (e.g. +0.1%), it is possible to observe the differences between traffic loads. This is because the system can cope with this arrival rate even it is more than the service rate. Packet loss performance of CP for the number of hotspot ports is depicted in Fig. 4.4. Equation 4.7 is used for this plot for number of hotspot ports k = {1, 2, 3, 4, 5} with traffic loads p = {0.85, 0.90, 0.95, 1.00}. Even though CP achieves a better performance when imbalanced input traffic is considered, packet loss rate increases together with the number of heavily loaded ports. The reason for this is that the number of congested queues increases together with the number of heavily loaded ports.

Developed numerical analysis indicates that CS has no control over queue lengths and therefore this scheme experiences dramatic performance degradations even when the degree of imbalance is marginal. On the other hand, CP allocates equal portions for each port and prevents heavily loaded ports from dominating the buffer memory. In other words, CP is fair to all output ports. Despite its protective nature and fairness,

performance degradation occurs as the number of hotspot ports increases. To conclude, an optimal buffer sharing scheme must benefit from (i) the flexibility of complete sharing, and (ii) the protective nature and fairness of complete partitioning.

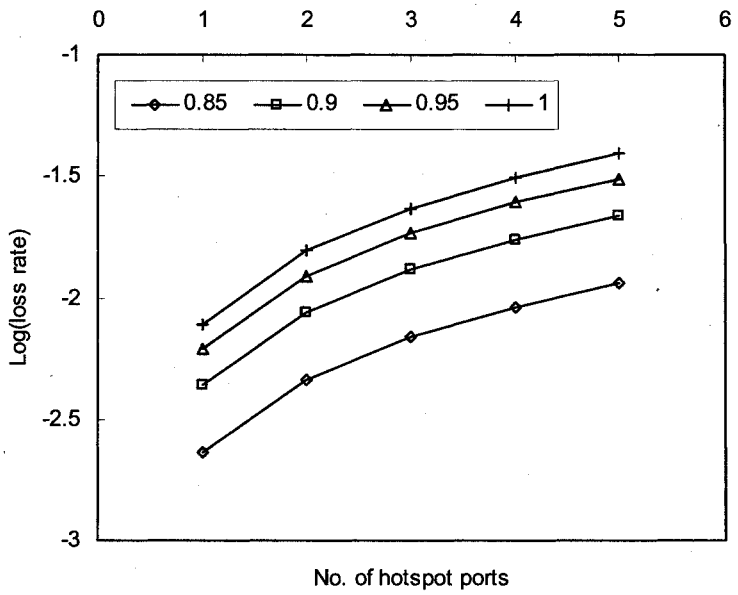


Fig. 4.4 – Complete partitioning: packet loss rate versus number of hotspot ports for various hotspot loads

4.4. Buffer Allocation Policies for Shared Buffer Packet Switches

Shared buffer switch performance is subject to high packet loss regardless of depth of its buffers. This is mainly because of imbalance and burstiness nature of the input traffic. Although complete partitioning allocation scheme partly resolves this issue, performance degradation is still possible, such that increase in the number of heavily loaded ports means more losses at the tail of heavily loaded queues. For this reason, different types of buffering schemes have been proposed in the literature in order to develop an optimal scheme. Buffer management in shared buffer switch architectures can be classified as pre-emptive and non-preemptive. Pre-emptive policies, such as

drop on demand (DoD) (Wei et al. 1991) and push-out with thresholds (POT) (Cidon et al. 1995; Sharma and Viniotis 1999), replace packets that are already in the buffers. Non-preemptive policies, also called threshold policies, reject packets before they actually enter the switch buffers. Threshold based policies are further classified as static thresholds (Irland 1978; Kamoun and Klienrock 1980), dynamic thresholds (Choudhury and Hahne 1998; Fan et al. 1999; Chu et al. 2002; Kesselman and Mansour 2002; Yang et al. 2003a), and adaptive control (Tipper and Sundareshan 1988; Ascia et al. 2002).

4.4.1. Static allocation policies

In this section, an overview of static policies, square-root rule, sharing with minimum allocation (SMA), sharing with maximum queue lengths (SMXQ), sharing with maximum queue length and minimum allocation (SMQMA), is presented. A performance analysis is then carried out to demonstrate the performance characteristics of these allocation policies.

4.4.1.1. static threshold

Square-root rule was proposed in a study by Irland (1978). This policy is a heuristic approximation of the optimal scheme. It is independent of the traffic load with fixed buffer limit. It was suggested that a fixed limit value can be chosen rather than imposing a dynamic scheme, which has to be re-optimised according to time-dependent factors, such as the traffic load and traffic pattern changes. Even though a fixed limit does not provide an optimal threshold value at all times, the shared buffer switch will be able to perform reasonably better than the complete sharing. Square-root rule can be formally presented as follows:

$$L = \frac{M}{\sqrt{N}}, \tag{4.8}$$

Where L denotes the limit rounded-up to the nearest integer, M denotes the buffer size in terms of the number of fixed-length packets and N denotes the number of input/output ports.

In sharing with minimum allocation (SMA) (Kamoun and Klienrock 1980), a minimum amount of buffer space is always reserved for each output port queue. SMA allows sharing a portion of buffer space completely and in addition a minimum amount of buffers b_i are permanently available to output port i at all times, $i = 1, 2, .. N$, where $b_i \leq M / N$. The domination caused by a heavily loaded port has a reduced impact on lightly loaded ports, as each output port can have access to a guaranteed buffer space. For this reason, SMA exploits a degree of fairness.

Sharing with maximum queue lengths (SMXQ) imposes a maximum queue length, which is greater than the fair share size M / N and less than the buffer size M (Kamoun and Klienrock 1980). The sum of these maximum queue lengths is greater than the buffer size:

$$\sum_{i=1}^{N} b_i \geq M, \tag{4.9}$$

Where b_i is the maximum size allocated to output port i. Although the above model sets separate maximum queue lengths for each output port, normally a single maximum value is used.

Sharing with maximum queue length and minimum allocation (SMQMA) combines both SMA and SMXQ schemes. It reserves dedicated buffers for each output port as well as limits the queue lengths by setting a maximum queue length. The advantage of SMQMA is that even though SMXQ restricts queue lengths, lightly loaded queues might be left with insufficient buffer space. Hence, SMQMA employs the SMA scheme to reserve a guaranteed buffer space for each output port.

4.4.1.2. numerical analysis

In this analysis, four of the static policies are considered, namely complete sharing, complete partitioning, square-root rule, and SMXQ. Both SMA and SMQMA schemes are excluded from this analysis, as they posses the same characteristics as SMXQ when the number of switch inputs/outputs is limited to two (e.g. 2×2) (Kamoun and Klienrock 1980). Performance analysis involves emulating the Markov state of an 2×2 shared buffer switch architecture (Fig. 4.5). A software program was developed for this analysis and the assumptions considered to model the simulation are as follows:

i. Switch model consists of two inputs and outputs $N = 2$ sharing a common finite buffer space M.
ii. Output ports remove a packet from the head of line of FIFO queues with deterministic service times.
iii. Network is synchronous (slotted) and at most one fixed-length packet is generated in a time-slot.
iv. Packets are generated by using independent identical Bernoulli process. Consecutive arrivals and arrivals to different output ports are independent of each other.

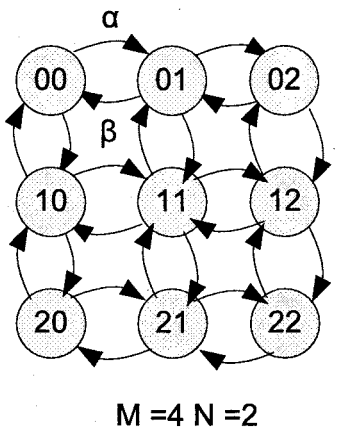


Fig. 4.5 – Markov state diagram for shared buffer switch model with two ports and buffer space with four packets

The main aim of the analysis is to observe the behaviour of each scheme rather than performance comparison. The metrics considered are the Markov state probability and the queue lengths over simulation time. Figure 4.6 illustrates transient plot of queue lengths (number of packets) versus simulation time (slots). The number of samples taken during this test run is 1000. Simulation starts with no packets in the system. In all four cases a steady state is reached in a short period of time with possible temporary fluctuations. In complete sharing scheme, shown in Fig. 4.6(a), port 1 hits the maximum allocation size (ten packets) due to highly imbalanced traffic. Packets destined to port 2 barely find buffer space due to high domination by port 1 packets. Maximum space that port 2 claims from the shared buffer is at most 20%. This is only possible when port 1 packets are not fully occupying the buffer space. Complete partition divides the buffer into two equal portions where port 1 queue length reaches a steady state of five packets (see Fig. 4.6(b)). It is possible to observe that port 2 is not fully utilising its dedicated buffer space. Square-root rule sets a maximum allocation, $\left(10/\sqrt{2}\right)$, rounded-up to the nearest integer. Hence, the queue length of any port queue cannot be longer than this maximum threshold. In this specific case, the

square-root rule spares a guaranteed 20% buffer space for the lightly loaded port (see Fig. 4.6(c)). In Fig. 4.6(d), SMXQ allocates buffer space just like the square-root rule. However, a predefined threshold parameter must be set. Maximum allowed queue length can be set as $T_i = \alpha.M$, where α denotes the threshold parameter. In this analysis threshold parameter is set as 0.7. For various traffic conditions this parameter has to be optimised through repeated trials.

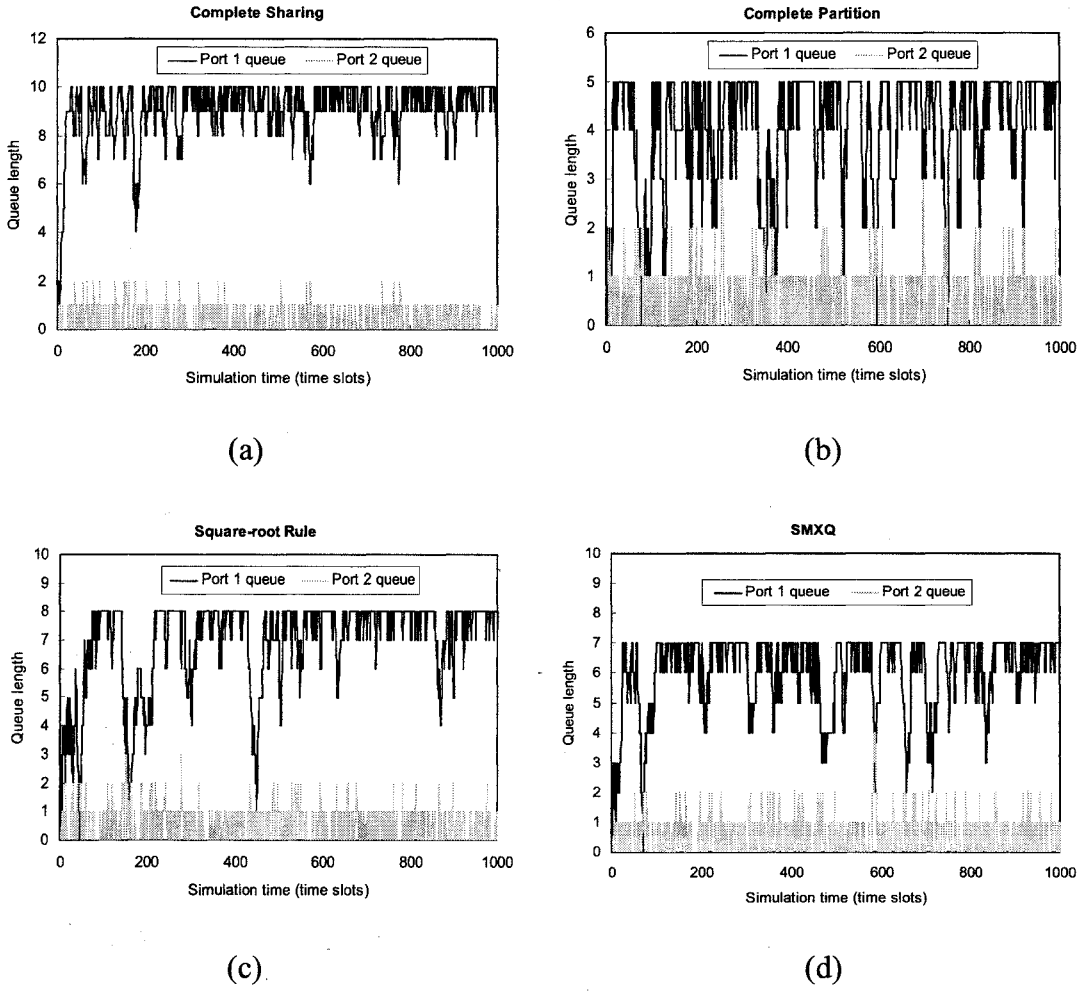


Fig. 4.6 – Queue length versus simulation time, (a) CS, (b) CP, (c) square-root rule, and (d) SMXQ (Traffic load = 0.8, port 1 load = 0.6, buffer size = 10)

State diagram probabilities are used to determine which states are more likely to be visited by each scheme under specific traffic conditions. The visited state probability is denoted as $\pi_{ij}(n)$. It is the probability that the allocation policy is in state at time n; note that the sum of state probabilities must equal to one and is formally represented as follows:

$$\pi_{ij}(n) = \Pr\{X_n = i, Y_n = j\} \quad \text{and} \sum_{i \times j} \pi_{ij}(n) = 1 \tag{4.10}$$

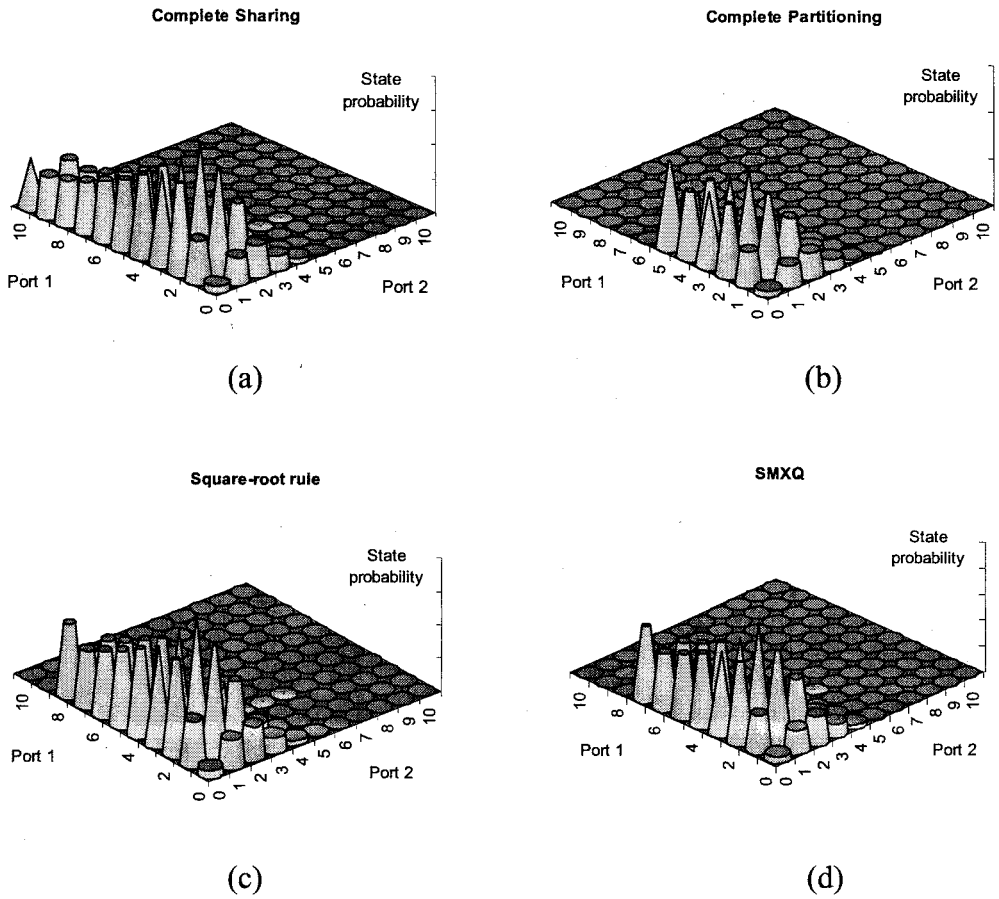


Fig. 4.7 – Markov state probabilities [port 1, port 2], (a) CS, (b) CP, (c) square-root rule, and (d) SMXQ (Traffic load = 0.8, port 1 load = 0.48, buffer size = 10)

In this specific case port 1 represents dimension i and port 2 represents dimension j, $[i, j]$. Parameter n is a non-negative integer, where $n \rightarrow (1\times10^6)$. Figure 4.7 shows state probabilities of the allocation schemes discussed in this analysis. Although the traffic is slightly imbalanced, the most likely states for complete sharing scheme are [i = 0..10, j = 0..2]. Complete partitioning restricts active queue lengths, therefore likely states are limited with [i = 0..5 and j = 0..2]. As can be seen from this figure, both square-root rule and SMXQ have a similar probability patterns. Only difference is that SMXQ has one layer of states less than square-root rule due to predefined parameter used by SMXQ. All schemes have a common state where state probability is at its highest, [$i = 2, j = 1$].

4.4.2. Dynamic threshold policies

4.4.2.1. dynamic thresholds (DT)

Dynamic threshold policy, proposed by Choudhury and Hahne, aims to achieve the simplicity of SMXQ together with the adaptivity of the pre-emptive policy (Choudhury and Hahne 1998). Simply, global threshold at time t, $T(t)$, is directly proportional to multiples of unused buffer space given as:

$$T(t) = \alpha.(M - \sum_{i=1}^{N} q_i(t)), \tag{4.11}$$

Where α is the control parameter, and $q_i(t)$ is the number of packets in the output port queue i at time t. Any queue that exceed the global threshold have their packets blocked temporarily, $q_i(t) > T(t)$. This scheme spares a small amount of buffer space for two reasons. Firstly, it allows inactive queues to have access to buffer space when they become active. Secondly, when the load on an active queue increases, it starts

acquiring from this unused space. Therefore, this action denotes that traffic conditions have changed and control is essential. The dynamic threshold policy is robust since it is sensitive to changes in the input traffic symmetry and phase. Whenever the load changes, the system goes through a transient state. For instance, when an inactive output port becomes active, this port will acquire more buffer space. Unused buffer space becomes smaller, hence the threshold, in order to free up more space for newly active queues. The main drawback of this policy is that granularity of allocation is bound to the control parameter. Control parameter α can be easily implemented by using a simple bit register with the condition of being multiples of two. For a very large α value, control is ineffective and very small α value is restrictive on queue lengths. In (Choudhury and Hahne 1998), it was shown that control parameter from 2^{-1} to 2^{1} offers optimal loss performance.

4.4.2.2. dynamic queue thresholds (DQT)

Dynamic queue thresholds policy, proposed by Fan et al., aims to satisfy the following dual objectives (Fan et al. 1999): (i) fair sharing and (ii) efficient utilisation of shared buffer space. When the switch is overloaded, each overloaded port should be able to access at least *M/N* buffer space. In addition to this, overloaded ports must have access to any unused buffer space to achieve maximum possible buffer utilisation. In order to satisfy the above goals, the following threshold based algorithm is used:

$$T_{new}(t) = \begin{cases} \max(q_i(t)+1, T_{old}), & if \quad \sum_{k=1}^{N} q_k(t) < \delta.M \\ \max(T_{old}-1, T_m), & if \quad \sum_{k=1}^{N} q_k(t) \geq \delta.M \end{cases} \tag{4.12}$$

Where $T_{new}(t)$ and $T_{old}(t)$ represent the new and old thresholds at time t respectively. Parameter δ denotes the share parameter ($0 < \delta \leq 1$) and T_m is the default threshold which is set statically. Share parameter close to one means complete sharing. Hence, $M - \delta.M$ is the guaranteed buffer space to be left aside for lightly loaded ports. Simply when a packet arrives to queue i at time t, new threshold is adjusted and queue length of port i is tested whether to admit or drop the incoming packet. Steady state representation of this scheme is given as:

$$T = \frac{\delta.M}{k_a} \tag{4.13}$$

Where k_a is the number of heavily loaded output port queues ($1 \leq k_a < N$).

4.4.2.3. partial sharing partial partitioning (PSPP)

Partial sharing partial partitioning policy, introduced by Chu et al., is a combination of compete sharing and complete partitioning control schemes (Chu et al. 2002). PSPP works in two modes: overloaded and non-overloaded. First of all, if a queue is possessing buffer space more than the fair share (M/N), then it is denoted as active queue. If there is at least one active queue, the switch is said to be in overloaded mode. When the switch is in overloaded mode, it uses a scheme which is a combination of complete sharing and complete partitioning. In non-overloaded mode, there is no control over buffers, i.e. complete sharing.

The main purpose of this allocation scheme is to spare buffer space for inactive queues and to adjust the global threshold for active queues accordingly. Determining a spare buffer space for inactive queues has the following motivations: (i) In case inactive queues become active, spared buffer space can be used to provide space for

incoming bursts; (ii) spared buffer space must be as small as possible to allow active queues to use unclaimed buffer space effectively to reduce unnecessary packet losses; finally, (iii) spared buffer space must adapt itself to changing traffic conditions. The following expression is used to determine the buffer space set aside for inactive queues:

$$M_{spare} = \beta.k(t), \tag{4.14}$$

Where $k(t)$ is the number of inactive queues at time t, and β denotes the threshold factor. A global threshold $T(t)$ is then set by using (4.14):

$$T(t) = \frac{M - \sum_{i \in s(t)} q_i(t) - \beta.k(t)}{N - k(t)} \tag{4.15}$$

Where $s(t)$ is the set of active queues at time t.

This global queue threshold is used to decide whether to admit or drop packets. If the queue length is greater than the threshold, packets are dropped; otherwise they are admitted to the switch buffers. In (Chu et al. 2002), it was suggested that for various traffic conditions, the optimal threshold factor value for PSPP ranges from 1.5 to 2.0. It is shown that when the PSPP threshold factor is below 1.5, packet loss rate increases. This is due to very small space being spared for lightly loaded ports. When the threshold factor is above 2.0, the buffer space reserved for lightly loaded ports is underutilised.

4.4.2.4. harmonic buffer management

Harmonic buffer management policy divides the memory between output ports according to their sorted size. The queue size of the *i*th port is proportional to $1/i$ of the memory (Kesselman and Mansour 2002). This scheme uses the harmonic series when allocating buffer space. For this reason, it is called the harmonic buffer management policy. In this policy the cumulative threshold for the total length of the k largest queues is given as:

$$B_k = \frac{M}{\ln(N)+1}\sum_{i=1}^{k} 1/i \,, \tag{4.16}$$

Where B_k is the cumulative threshold and k is the number of large queues. Hence, the following inequality is checked for k largest queues when admitting packets:

$$\sum_{i=1}^{k} q_{s(i)} \le B_k \qquad : k = 1...N \tag{4.17}$$

Where $q_{s(i)}$ denotes the length of the queue i, $s(i)$ is the ith largest queue (in sorted order).

This scheme allocates up to $\frac{M}{\ln(N)+1}$ packets and reserves sufficient buffer space for lightly loaded queues, which is given as $\frac{1}{N}\frac{M}{\ln(N)+1}$. It is assumed that $M > N.\ln(N)$.

Note that when $k = N$, the cumulative threshold is less than the buffer memory size. Results given in (Kesselman and Mansour 2002) suggest that this policy performs well under general traffic conditions. The only drawback of this scheme is the operation complexity. A track of active queues and their relative lengths must be kept.

Queue lengths must be checked on each packet arrival or batch arrivals and queues must be re-sorted accordingly.

4.4.2.5. threshold based selective drop (TSD)

Threshold based selective drop scheme is based on the max-min bandwidth sharing scheme (Yang et al. 2003a). It is assumed that all congested queues, where queue lengths are greater than the threshold, have the same queue lengths at time t. This length is called the fair share size $\alpha(t)$. Each output port acquires a buffer space given by $\min\{q_i(t), \alpha(t)\}$. The TSD scheme checks two conditions when admitting and rejecting packets:

i. If $\sum_{i=1}^{N} q_i \geq (M - T)$, then the buffer is congested and TSD factor T is used to achieve fair size among congested output ports. Fair share size is derived from the following expression:

$$\sum_{i=1}^{N} \min\{q_i(t), \alpha(t)\} = M - T \tag{4.18}$$

Where M and N denote the total buffer and switch sizes, respectively.

ii. Otherwise, the switch is not congested and hence allocated fair buffer size is given as follows:

$$\alpha(t) = \max_i(q_i(t)) \tag{4.19}$$

The results given in (Yang et al. 2003a) suggested that TSD scheme achieves performance close to a pre-emptive policy. Also, it was shown that optimal loss can be achieved when the value for TSD factor is set as 16.

4.4.3. Pre-emptive policies

4.4.3.1. drop on demand (DoD)

When switch buffers are saturated, drop on demand policy, also known as push out (PO), either drops the incoming packet or replaces the one which is already in the buffer memory. Drop on demand policy is based on the following objectives (Wei et al. 1991):

i. There is no limit on the amount of buffer space allocated to each output port.
ii. If there is sufficient space in buffers, incoming packets are admitted.
iii. A packet arrives at queue of output port *i* and finds the buffer full. In such case, arriving packet pushes out a packet from the head of the longest queue. If port *i* queue is the longest queue, then arriving packet is rejected. Otherwise, port *i* packet is accepted to join the tail of output port queue *i* and a packet is removed from the head of line of the longest queue.

A similar simulation setup is used as in Section 4.4.1 to observe the queue lengths and state probabilities. Transient plot for DoD is shown in Fig. 4.8. Like complete sharing, the DoD scheme makes better use of buffers. When buffers are not used by lightly loaded port (i.e. Port 2), they are fully occupied by packets destined to port 1. These buffers are released when port 2 packets arrive. State diagram of DoD is depicted in Fig. 4.9. Although its state diagram is similar to complete sharing, there is a big difference among these schemes in terms of throughput performance (see Fig. 4.10). The main reason for this performance implication is that the DoD policy differs from the complete sharing in state probability $\pi_{ij}(n) = \Pr\{X_n = 10, Y_n = 0\}$. The DoD policy is less likely to be in this state. When buffer memory is congested and a port 2

packet requires buffer space on arrival, port 1 packet is pushed out of the queue. Hence, the switch state moves to probability $\pi_{ij}(n) = \Pr\{X_n = 9, Y_n = 1\}$. On the other hand, complete sharing stays in the same state in a similar situation blocking arrivals to port 2. In Fig. 4.10, switch buffers cannot accommodate traffic loads when $p > 0.7$. The DoD scheme is able to provide improved performance over complete sharing due to its purging mechanism.

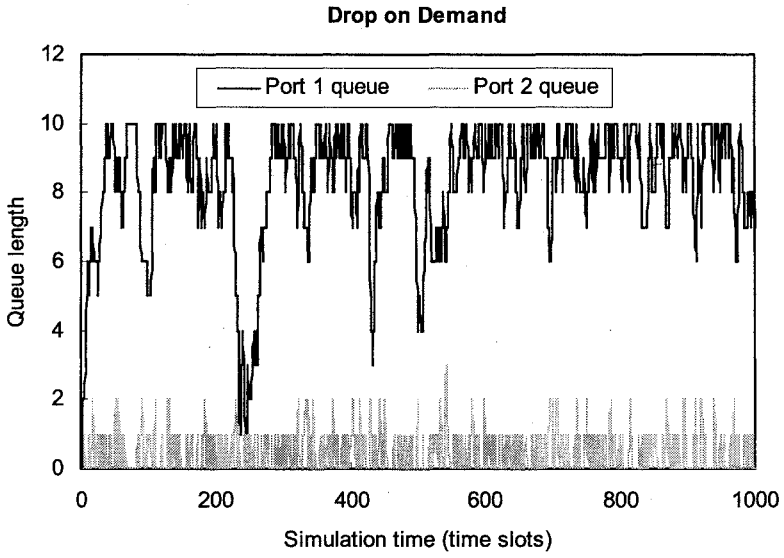


Fig. 4.8 – DoD transient plot

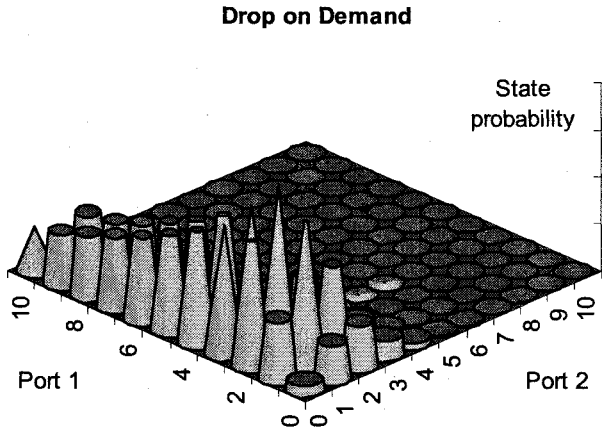


Fig. 4.9 – DoD state probabilities

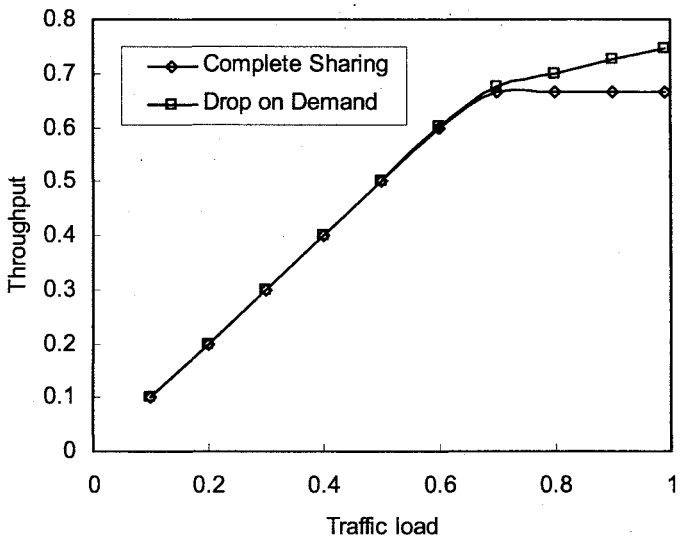


Fig. 4.10 – DoD throughput performance
(Port 1 load = 0.75 × Traffic load (p); buffer size = 100)

The DoD policy has the following distinctive characteristics (Sharma and Viniotis 1999): (i) Fairness - allows smaller queue to grow by claiming buffer space from lengthy queues, (ii) efficiency – avoids output-idling, (iii) resource utilisation – packets are stored until when the loss is unavoidable and buffer utilisation close to 100% can be achieved in bursty traffic conditions, (iv) naturally adaptive - active queues are allowed to grow unless lightly loaded queues claim buffer space. Nevertheless, this superb performance comes with a number of drawbacks. Firstly, purging a packet which is already in the buffer memory is a costly process. It is much easier to reject a packet before it enters the buffers. Secondly, switch must monitor the lengths of each output port queue. In an $N \times N$ switch searching for the longest queue has a complexity $O(N)$. As $N \rightarrow \infty$, it is impossible to find the longest queue and therefore nearly longest queue is considered. For this reason performance degradation is very likely. According to the above claims, it is possible to conclude that it is almost impossible to implement this buffer allocation policy.

4.4.3.2. push-out with thresholds (POT)

Push-out with thresholds (POT), also known as complete sharing with virtual partition (CSVP), was considered in two independent studies by Wu and Mark (1995) and Cidon et al. (1995). This buffer allocation policy is a pre-emptive policy where an over-subscribed port queue is forced to purge its head of line packet if a pre-defined threshold is exceeded.

A virtual partition is a buffer segment which is allocated to a specific port (user). The total available buffer space M is partitioned into N segments with sizes b_1, b_2... and b_N, such that $\sum_{i=1}^{N} b_i = M$. Allocation policy works in two modes; buffer is full and buffer is not full. When buffer is not full, complete sharing is employed. Packets destined to any of the output ports are admitted to the buffer. When the buffer is full, if queue length of port i is less than the partition size $q_i < b_i$ at least one other queue must be occupying more than its partition size. In this case, an over-subscribed queue packet is pushed out from the buffer and replaced with the newly arrived port i packet. However, if port i queue is occupying more than its partition size, newly arrived packet is rejected. It is not necessary to partition virtual segments uniformly. These can be defined arbitrarily according to estimated traffic loading conditions.

In (Wu and Mark 1995), it was shown that the CSVP scheme achieves better performance than the complete partitioning and is fair to individual ports (or users). When the input traffic is heavy, CSVP is characterised as complete partition. In a similar study, Cidon et al. has proved that the push-out with thresholds is an optimal policy for a system with two inputs and outputs (Cidon et al. 1995). It was also shown that push-out policy is not optimal in case of asymmetric input loads and is optimal

only for symmetric traffic. It was argued that push-out with threshold policy has very little performance improvements (in terms of packet loss) when compared to an optimal coordinate-convex policy. Coordinate-convex policies are those policies that never pre-empt a packet once it is in the buffer memory (e.g. static thresholds, dynamic thresholds). A coordinate-convex state space, Ω, is such that if $n = (n_1, n_2, ..., n_N) \in \Omega$, then $n_i^+ = (n_1, n_2, ..., (n_i - 1), ..., n_N) \in \Omega$ for all i = 1, 2..., N. This mathematical representation suggests that a departure from this state is never blocked. Based on these findings one can easily state that employing an optimal coordinate-convex policy is more feasible than a pre-emptive policy. This is because implementation of push-out policies is rather impractical considering the performance gain over an optimal coordinate-convex policy (Cidon et al. 1995).

4.4.3.3. push-out with virtual thresholds (PVT)

Push-out with virtual thresholds is a pre-emptive scheme that supports different loss probabilities for various priority packets. This policy was proposed to provide support for priority traffic and overcome the problems encountered by the selective push-out (SP) scheme (Yang et al. 2003b).

The SP scheme is normally applied to the shared buffer. It considers two traffic classes, 0 and 1 (Class 0 being the high priority class). As any other push-out scheme, when the buffer is not full, all arrivals are admitted to the buffer memory. When the buffer is full, the arriving Class 0 packet could push out a Class 1 packet near to the head of line of the longest queue. Note that different traffic classes destined for the same output port are admitted in the same FIFO queue in the order of arrival. If there

are no Class 1 packets in the shared buffer, a Class 0 packet is removed from the head of line of the longest queue. Arriving Class 1 packet is allowed to remove another Class 1 packet from the longest queue. If there are no Class 1 packets, then arriving Class 1 packets is rejected. This scheme achieves relatively low loss performance for Class 0 packets. This scheme is not fair, as it removes Class 1 packets from the buffer immensely. For this reason, overall loss performance might be degraded.

PVT is considered as a combination of thresholds and selective push-out. Virtual threshold (VT) is determined according to fairness, Class 1 packet loss probability and overall packet loss probability (Yang et al. 2003b). Very large VT can be employed, if fairness and overall loss performance are the main considerations. On the other hand, very small VT is used if low loss is desired. PVT is characterised as follows. When the buffer is not full, all arrivals are admitted. Otherwise, an arriving Class 0 packet can push-out a Class 1 packet closest to the head of the longest queue with length greater than its predefined VT. If there are no Class 1 packets, Class 0 packet pushes out the head of line of the longest queue. If this packet is destined to the longest queue, then it is rejected. When a Class 1 packet arrives at a queue whose length is smaller than its VT, it pushes out a Class 1 packet whose queue length is greater than the queue VT. In the same context if there are no Class 1 packets, Class 0 packet from the longest queue is pushed out. When a Class 1 packet arrives at a queue whose length is greater than its VT, it pushes out a Class 1 packet whose queue length is greater than queue VT. In this context, if there are no Class 1 packets, arriving packet is simply rejected. The performance implications of push-out with thresholds scheme is presented in (Yang et al. 2003b) via a simulation study. The results show

that for different set of requirements, such as fairness and overall loss, VT value must be adjusted accordingly.

4.4.4. Adaptive control

4.4.4.1. optimising control

Optimising control scheme is inspired from dynamic allocation of buffers to individual output ports by using a mathematical optimisation framework (Tipper and Sundareshan 1988). This optimisation framework aims to maximise the nodal throughput. This policy states that there exists a vector of buffer allocations, $b = [b_1, b_2 \ldots, b_i \ldots, b_N]$, such that the normalised throughput is at an optimal or sub-optimal level. Throughput is repeatedly maximised despite the changes in traffic arrivals. Queues are treated independently as arrivals and departures are independent for each queue. This can be represented as a weighted sum of the normalised throughputs of individual output ports. Hence, the objective function can be formally represented as follows:

$$Minimize \quad \sum_{i=1}^{N} w_i p_i BP_i \tag{4.20}$$

Where w_i is the weighting constant of output port i, p_i denotes the utilisation of output port i, $p_i = \lambda_i / \mu_i$ (λ and μ denote arrival and departure rates respectively), and BP_i is the blocking probability of output port i.

Authors in (Tipper and Sundareshan 1988) argued that in order to be able to solve the optimisation problem, performance metrics must be represented by using a closed form expression, such as queuing model. This helps to define the problem domain

more precisely. Each queue is represented as an $M/M/1/b_i$ queue and blocking probability for this queuing model is formally given as:

$$Minimize \quad \sum_{i=1}^{N} w_i p_i \left[\frac{(1-p_i)p_i^{b_i}}{1-p_i^{b_i+1}} \right] \tag{4.21}$$

Arrival phase to an output port is determined by using estimation methods. Buffer allocation vector b is given by solving the above problem.

Authors in (Tipper and Sundareshan 1988) also suggested a number of solution techniques for this problem. One possible solution is to visit all possible states in problem space by using an exhaustive search and choose the one with optimal objective function value. This technique is not ideal considering the number of possible states (e.g. N inputs/outputs and buffer size M) and also the dynamic load environment. Other solution techniques involve using nonlinear programming algorithms and the branch and bound technique. Although the suggested optimisation framework is fully dynamic, it has a number of shortcomings. For example, Poisson arrival/departure is rather unrealistic. Moreover system is very dynamic and problem state is changing continuously. That is, while searching the optimal state b^*, this state might no longer be optimal at time t^+. Also computation complexity of an optimal solution is relatively high when compared with dynamic threshold schemes.

4.4.4.2. integrated fuzzy-GA

This technique employs two well-known artificial intelligence techniques, the fuzzy logic and the genetic algorithms, to employ an optimal switch control (Ascia et al. 2002). Fuzzy logic is used for adaptive control. It uses a set of fuzzy rules that

determine the relationship between system variables. In a slotted system, at each time-slot control threshold is updated by using the inference rules. Based on the overall buffer occupation, rule base is used to select the threshold. For instance, if the buffer occupation is very high, threshold decreases accordingly. Membership functions determine the relationship between occupancy and threshold control. Genetic algorithms is used to deduce the fuzzy knowledge base according to the given buffer and switch size parameters. Genetic algorithms works off-line. It uses a fitness function to evaluate the fitness of each solution. Selection, crossover and mutation operations are carried out repeatedly until a terminating condition is met. Finally, derived fuzzy knowledge is used online to determine the global threshold dynamically.

The results in (Ascia et al. 2002) suggest that integrated fuzzy-GA scheme outperforms the well-known dynamic thresholds described in (Choudhury and Hahne 1998) and achieves performances close to DoD policy. The genetic algorithms is used off-line in order to form the rule base. The control latency (threshold computation + control) introduced by fuzzy control is as low as 1% of Gb/s line speed rates (Ascia et al. 2002). Thus, fuzzy-GA scheme does not retain any implementation complexities and hardware realisation is rather tractable.

4.5. Summary

In this chapter, the effect of asymmetric input traffic on shared buffer packet switch was studied by means of numerical analysis. Results show that there is a major difference when buffer space is completely shared and when equally portioned among

output ports. Numerous techniques have been proposed in the literature to overcome the performance degradation due to imbalanced input traffic. These are mainly categorised as static threshold, dynamic threshold, adaptive control and pre-emptive. Unlike static threshold policies, dynamic threshold policies can cope with changing traffic phase and adopt thresholds accordingly. Adaptive control schemes adopt an optimal threshold policy over time. Finally, pre-emptive policies replace a packet which is already in the switch buffers. These achieve the best loss performance, but practical implementation is almost impossible.

CHAPTER 5

DECAY FUNCTION THRESHOLD POLICY FOR OPTIMAL BUFFER SHARING

5.1. Introduction

In this chapter an improved threshold based dynamic buffer allocation policy for shared buffer switch architectures, known as decay function threshold (DFT) policy, is developed and studied. Proposed scheme uses two environment variables to effectively realise the changes in network dynamics. It is simple to implement and supports priority traffic.

The next section discusses the motives for developing a better scheme and identifies a series of objectives for optimal sharing. Section 5.3 describes the proposed scheme. In this section two versions of this policy are devised. In Section 5.4, steady state analysis is carried out to demonstrate how queue lengths converge to a steady-state allocation. Finally, in Section 5.5, an integration framework for buffer management and bandwidth scheduling is presented.

5.2. Motivation and Challenges

Shared buffer switch architectures offer improved performance when compared with other buffering schemes, such as input queueing and output queueing, due to its ability to allocate buffer resources in a more flexible manner. Nevertheless, a buffer management scheme is required when the input traffic is not uniformly distributed

across the output ports. Even though the input traffic is uniformly distributed and it is non-bursty, temporary fluctuations may cause congestion on a particular output port. As a result, a high volume of traffic is destined to one or more output ports and common buffer space is distributed unfairly among output ports.

Static threshold policies are simple to implement due to their blocking nature. Static thresholds must be tuned statistically to achieve an optimal packet loss performance. In a highly dynamic network environment static policies fail to match the traffic phase, thus the network requirements. Preemptive policies can match the input traffic phase as they are naturally adaptive. Packet losses occur only when the loss is unavoidable. Despite the performance offered by this type of policies, preemptive policies are not easy to implement. For this reason dynamic threshold policies are the preferred candidates since they are blocking type and also have the capability to adapt to changes occurring within traffic. However, there are a number of issues that need to be addressed. An optimal buffer allocation scheme must exploit the following features:

- *Fairness*: The main feature of a dynamic buffer management scheme is to prevent idling of output ports. This is only possible by allocating sufficient buffers for each output port, e.g. complete partitioning scheme is fair, as it allocates equal partition for each output port.
- *Robustness*: Regardless of the traffic conditions, the control scheme must be able to offer good performance with marginal tuning of the policy.
- *Scalability*: The computation of the optimal threshold value in a switch with many inputs and outputs is inherently a difficult task.

- *Blocking nature*: Amount of buffers available to an output port must be restricted and packets must be blocked on arrival.
- *Sensitive to traffic phase*: Packet admission and rejection decision is made on every arrival and it is independent of previous or future decisions.
- *Utilise environment variables*: Environment variables, such as traffic intensity and imbalance, packet inter-arrival times, buffer state, number of heavily loaded output ports, arrival rate of a specific traffic stream (e.g. priority traffic) etc., expose the current state of the network at a specific instance.
- *Ease of implementation*: There is a trade off between achieving a high performance and implementation complexity. Control logic is implemented by using gate logic and bundled with other hardware into a single integrated circuit (e.g. switch-on-chips). In other words, it has to be cost-effective.
- *Operation overhead*: Packet admission and rejection decisions must be made in a very short period of time. Note that in an $N \times N$ switch, control logic must be able to make decision N times the line speed. Otherwise a backlog of buffer space requests makes the switch non-functional.
- *Support for priority traffic*: Buffer allocation policies must be able to work hand-in-hand with a bandwidth scheduler to differentiate services between different classes of traffic.

Ideally, a buffer allocation policy must have the performance of a preemptive policy and simplicity of a static threshold policy. Dynamic thresholds policy (proposed by Choudhury and Hahne (1998)) offers this to an extent, but due to the multiplicative nature of share parameter α used by this scheme, the granularity of allocation is

somewhat coarse. As a consequence, the robustness of this scheme is rather controversial (e.g. for a particular condition the share parameter is optimal, but not always). Designing a fully adaptive buffer allocation policy by using machine intelligence techniques is attractive, but the operation overhead might be seen as a shortcoming. Possibly this is why adaptive control schemes have not gained much popularity. Recent studies in computational intelligence have shown that inherently difficult and ill-defined problems can be solved in an efficient effective manner by employing machine learning techniques, such as reinforcement learning (Boyan and Littman 1994; Ferra et al. 2003), stochastic learning automata (Sutton and Barto 1998), fuzzy logic (Ascia et al. 2002), heuristics etc.

The next section presents the proposed dynamic buffer allocation policy, the decay function threshold policy. This policy aims to satisfy most of the objectives discussed in this section and offer an improved performance over existing dynamic thresholds policies.

5.3. Decay Function Threshold Policy

Decay function threshold policy attempts to limit output port queue lengths by employing a control threshold. In this section, this control threshold is derived systematically. Let's make the following definitions:

i. A queue is said to be active if its length is longer than the fair share size, M/N, where M and N denote the buffer size and number of ports respectively.

ii. Threshold $T(t)$ is a global threshold applied to all queues at time t and all arrivals to queue i is blocked when $q_i(t) > T(t)$.

iii. Threshold $T_i(t)$ is a per-queue threshold tailored for queue i at time t. All arrivals to queue i is blocked when $q_i(t) > T_i(t)$.

In order to employ control threshold on an active queue, available buffer size can be divided into l equal partitions, $T = M/l$. For instance, when $l = N$, each output receives an equal portion from the shared buffer space. To achieve a dynamic scheme, l must be dependent on dynamic variables such as available buffer space, number of active queues, traffic load etc., and also there exists an optimal value for l such that loss performance is at its minimum level. Hence, dynamic threshold can be formally given as:

$$T(t) = \frac{M}{l(t)}. \tag{5.1}$$

If $l(t)$ is of integer type, the threshold given by the scheme is rather coarse. Therefore this parameter must be of floating point number type.

Based on observations, a number of postulations are made:

Postulation 1: "The buffer space that an output port has access decays exponentially together with its queue length".

Based on the above claim variable $l(t)$ can be formally given as:

$$l(t) = c^{\left\lfloor \frac{N.q_i(t)}{M} \right\rfloor}, \tag{5.2}$$

Where $q_i(t)$ represents the queue length of port i at time t. Parameter c determines the degree of sharing. For a very large c, the threshold scheme becomes more restrictive.

Parameters M and N help to adjust the control threshold according to the switch and buffer sizes. One can state that this scheme is less sensitive to network conditions. Therefore, an additional variable can be introduced to the above equation to better realise the changes in network.

Postulation 2: "The buffer space that an output port accesses decays exponentially together with its queue length and total buffer occupancy".

This claim can be formally represented as:

$$l(t) = c^{\left\lfloor \frac{q_i(t)}{B+1} \right\rfloor}, \tag{5.3}$$

Where B denotes the empty buffer space.

The ratio of queue size of queue i (e.g. number of packets) and empty buffer space, $q_i/(B+1)$, determines how active a queue is in contrast to unused buffer space. The aim of this scheme is to allow smaller queues to grow even when the buffer space is at its critical level. Using equations 5.2 and 5.3 and substituting into Equation 5.1, the following two threshold policies can be derived. These are named as DFT1 and DFT2:

$$DFT1: \quad T_i(t) = \frac{M}{c^{\left\lfloor \frac{N.q_i(t)}{M} \right\rfloor}}, \tag{5.4}$$

$$DFT2: \quad T_i(t) = \frac{M}{c^{\left\lfloor \frac{q_i(t)}{B+1} \right\rfloor}}. \tag{5.5}$$

Both policies offer tailored thresholds for individual queues. DFT1 employs a single parameter (queue length) to determine the threshold. Whereas DFT2 policy employs two variables, individual queue length and unused buffer space. The implications of queue length and empty buffer space on thresholds for DFT2 policy is illustrated in Fig. 5.1. It is possible to observe that when the available buffer space is very low, small queues can still benefit and accept incoming packets as long as buffer is not fully occupied. Longer queues have their packets blocked until the available buffer space becomes sufficiently enough for smaller queues to grow. In both equations, right hand side of the equation produces a floating point number, which is then rounded-up to the nearest integer. This is because the number of fixed-length packets is considered when calculating the queue lengths.

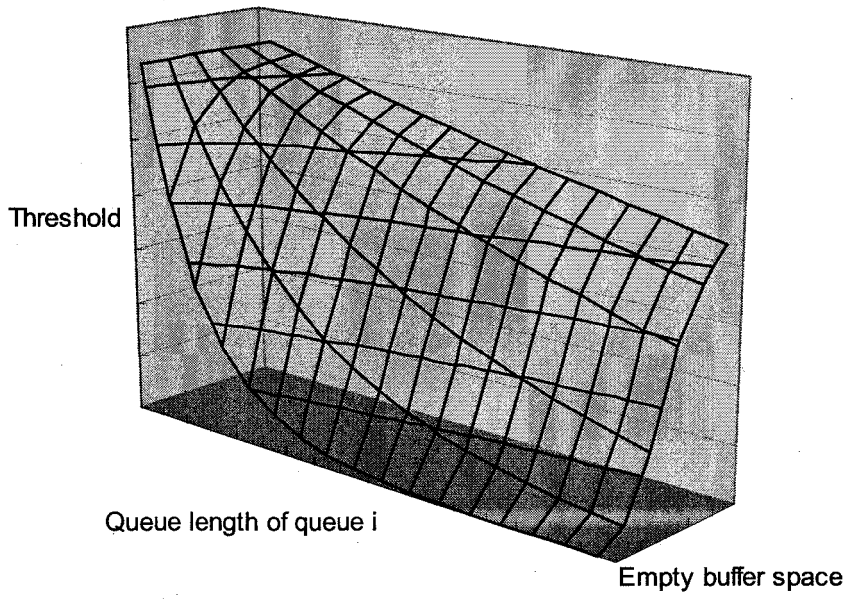


Fig. 5.1 – The effect of queue length and empty buffer space on thresholds (DFT2)

In both of the policies, the optimal value of parameter *c* can be achieved by observing switch performance under various network conditions, such as the traffic load,

number of heavily loaded ports and load on heavily loaded ports. Nevertheless, it is possible to carry out a numerical analysis to determine a coarse value for this parameter. A finer tuning might be needed depending on the environment requirements. In order to illustrate the effect of choice of parameter *c*, let's consider the DFT2 policy. One of these policies was chosen to avoid repeated analysis. Besides, DFT2 can be considered as more established form of the DFT1 policy, as it makes better use of environment dynamics.

In shared buffer switches, the following equation must be satisfied at all times:

$$M = \sum_{i=1}^{s} q_i(t) + \eta + B(t), \qquad (s < N) \qquad (5.6)$$

Where s denotes the number of heavily loaded ports, and η is the buffer space occupied by lightly loaded ports. To simplify this analysis, let's assume that all active queues receive exactly the same amount of input traffic and they don't interfere with each other for buffer space. Hence all heavily loaded ports acquire approximately the same amount of buffer space and they are under the control of similar thresholds (i.e. active queue lengths are the same as the threshold). Hence, Equation 5.6 can be rewritten as:

$$M = s.T + \eta + B, \qquad (5.7)$$

Substituting (5.5) into (5.7) and solving for $q_i(t)$ gives the following steady-state equivalent formulae for DFT2:

$$q_i \cong T = -\log_c\left(\frac{M - B - \eta}{s.M}\right).(B+1). \qquad (5.8)$$

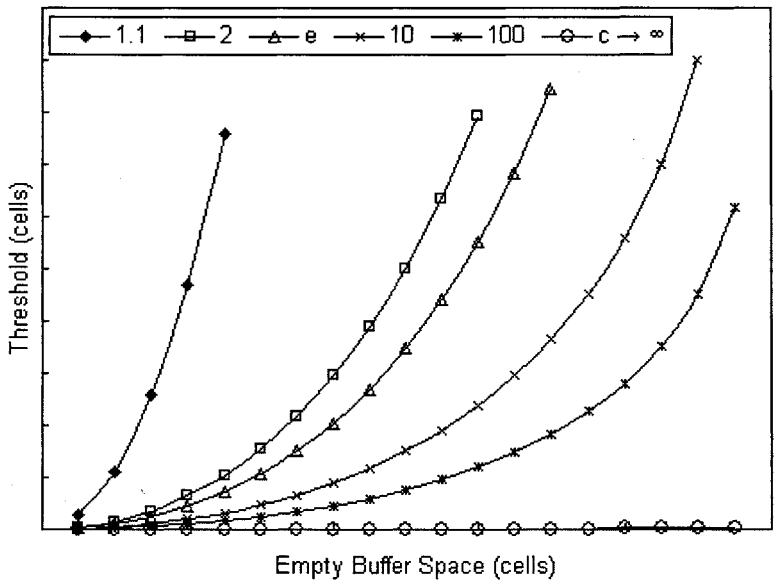


Fig. 5.2 – Threshold against the empty space for a range of c values

Equation 5.8 can only be satisfied when there exists at least one heavily loaded port where $s > 0$ and $0 < (M - B - \eta)/(s.M) < 1$ and finally $c > 1$. To determine a coarse value for c, threshold (T) versus empty buffer space (B) for c = {1.1, 2, e, 10, 100} and $c \rightarrow \infty$ is observed by using Equation 5.8. Results for this analysis are depicted in Fig. 5.2. As can be seen from this figure, parameter c determines the degree of control over heavily loaded ports. As $c \rightarrow \infty$, control threshold becomes more restrictive on the queue lengths. On the contrary control policy becomes more like complete sharing as c approaches one. A finer tuning is required for parameter c depending on the pattern and volume of input traffic, and degree of imbalance. For this reason, an empirical study is carried out later in this thesis to determine an optimal value for c.

5.4. Transient Analysis

Transient analysis demonstrates the behaviour of the DFT policy. For this analysis a similar methodology is followed as stated in (Choudhury and Hahne 1998; Chu et al.

2002; Yang et al. 2003a). This analysis illustrates how queue lengths of active queues and newly active queues converge to a steady-state allocation given by Equation 5.8. For instance, assume that $\eta = 0$ (i.e. $N - s$ inactive output ports), and by using (5.8) threshold at $t = 0$ is rewritten as:

$$T(0) = -\log_c(\frac{M-B}{s.M}).(B+1). \tag{5.9}$$

The queue length for s active and k inactive queues at $t = 0$ is given as:

$$q_i(0) = \begin{cases} -\log_c(\frac{M-B}{s.M}).(B+1), & for \quad 0 \le i < s \\ 0, & for \quad s \le i < s+k \end{cases}. \tag{5.10}$$

Simultaneously, k inactive queues become active with rate r - 1 and start acquiring buffer space. Queue length rate of change is bounded below by -1 if queue length is greater than the threshold, otherwise $r - 1$. That is, all arrivals to s active queues are rejected with rate -1 and arrivals to newly active queues are rejected with rate $r - 1$.

$$q_i(t) = \begin{cases} -1, & if \quad q_i(t) > T(t) \\ r-1, & if \quad q_i(t) < T(t) \end{cases}. \tag{5.11}$$

At this stage analysis is divided into two cases according to the value of r. In Case 1, the following condition is assumed for r:

$$r \le 1 + \frac{s}{k}. \tag{5.12}$$

This means that s initially active queues track the dropping threshold by discarding portion of their arrivals. Hence:

$$T(0+) = \frac{-k.(r-1)}{s} \quad \text{for } t < t_1, \tag{5.13}$$

Where t_1 is the time when the threshold becomes stable. Therefore rate of change in queue sizes for both already active and newly active queues is as follows:

$$q_i(0+) = \begin{cases} \dfrac{-k.(r-1)}{s} & for \quad 0 \le i < s \\ r-1 & for \quad s \le i < k \end{cases}. \tag{5.14}$$

Equations 5.13 and 5.14 hold as long as $t < t_1$. Therefore from initial conditions (5.10), the following threshold and queue lengths are derived:

$$T(t) = -\log_c(\frac{M-B}{s.M}).(B+1) - \frac{k.(r-1)}{s}.t \tag{5.15}$$

$$q_i(t) = \begin{cases} -\log_c(\dfrac{M-B}{s.M}).(B+1) - \dfrac{k.(r-1)}{s}.t & for \quad 0 \le i < s \\ (r-1).t & for \quad S \le i < k \end{cases} \tag{5.16}$$

Equation 5.16 is valid until the time t_1 when k newly active queues hit the threshold. Equating 5.15 and second part of (5.16) and solving for t_1 gives (5.17) when $t = t_1$:

$$t_1 = \frac{-\log_c(\dfrac{M-B}{s.M}).(B+1).s}{(k+s).(r-1)}. \tag{5.17}$$

Finally at time t_1, threshold $T(t_1)$ becomes:

$$q_i(t_1) = T(t_1) = -\log_c(\frac{M-B}{s.M}).(B+1).(\frac{k}{k+s}) \qquad \text{for } 0 \le i < s+k \qquad (5.18)$$

For $t > t_1$, both the queue lengths and threshold reach equilibrium and each queue maintains a constant length by being served at rate one, admitting arrivals at rate one, and rejecting arrivals at rate r - 1.

In Case 2, it is assumed that:

$$r > 1 + \frac{s}{k} \qquad (5.19)$$

From Equation 5.19 one can state that it is impossible for s already active queues to track the dropping threshold. In such a case, they will block all their arrivals and the queue lengths will only decline with rate -1:

$$T(0+) = \frac{-k.(r-1)}{s} \qquad \text{for } t < t_2 \qquad (5.20)$$

$$q_i(0+) = \begin{cases} -1 & for \quad 0 \le i < s \\ r-1 & for \quad s \le i < k \end{cases}. \qquad (5.21)$$

Threshold at $0 < t < t_2$ is given as:

$$T(t) = -\log_c(\frac{M-B}{s.M}).(B+1) - \frac{k.(r-1).t}{s}. \qquad (5.22)$$

In the same way queue lengths are given as:

$$q_i(t) = \begin{cases} -\log_c(\frac{M-B}{s.M}).(B+1) - t & for \quad 0 \le i < s \\ \frac{s}{k}.t & for \quad s \le i < k \end{cases} \qquad (5.23)$$

Equations 5.22 and 5.23 are valid as long as $0 < t < t_2$ where t_2 is the instance when the k new queues hit the threshold. Equating 5.22 and second part of (5.23) and solving for $t = t_2$ yields:

$$t_2 = \frac{-\log_c(\frac{M-B}{s.M}).(B+1).k}{(k+s)}. \tag{5.24}$$

At t_2 the threshold and queue lengths are as follows:

$$q_i(t_2) = T(t_2) = -\log_c(\frac{M-B}{s.M}).(B+1).(\frac{s}{k+s}) \quad \text{for } 0 \le i < s+k \tag{5.25}$$

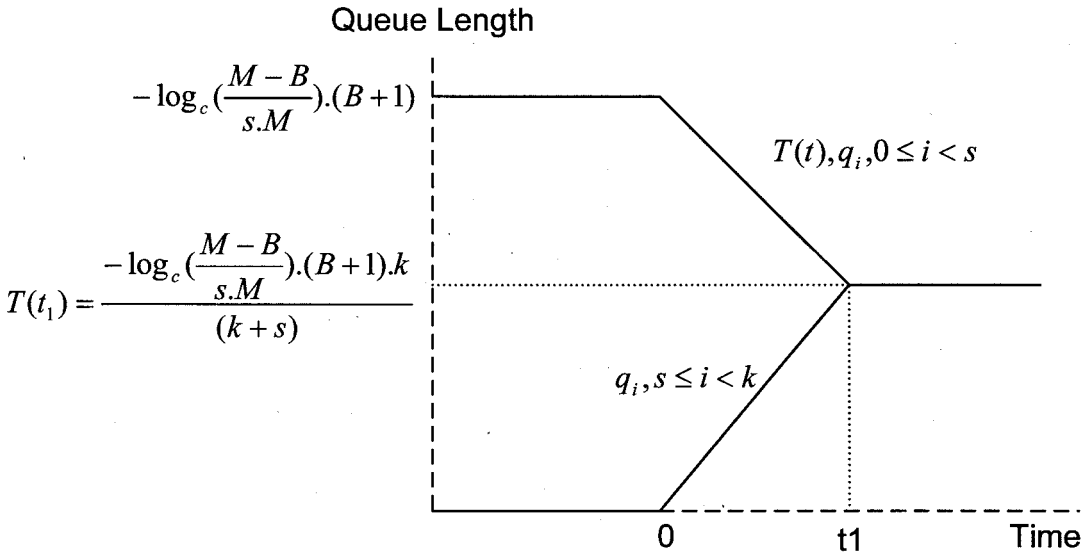


Fig. 5.3 – Transient behaviour of s initially active queues that track the dropping threshold by discarding portion of their arrivals

The difference between Equations 5.18 and 5.25 shows that Equation 5.25 is affected the most due to increase in the number of newly active queues. However, Equation 5.18 suffers the least in increase in the number of active queues. For very large k, there is a small change in the threshold in Case 1 (see Fig. 5.3). On the other hand, in

Case 2, the threshold becomes very small immediately to spare sufficient buffer space for k newly active queues. In both cases once a steady state allocation is reached, each queue blocks just enough traffic to maintain stabilised queue lengths. For very small c value, the performance of newly active queues can be very poor, as the threshold control cannot limit the misbehaving queues to set sufficient buffer space for newly active queues.

In this section transient behaviour of the DFT policy is presented. This analysis has provided us with better understanding of how the threshold policy behaves when new queues start acquiring buffer space. Besides the choice of parameter c is crucial for providing a tailored buffer allocation for newly active queues during transition.

5.5. An Integration Framework for Buffer Management and Scheduling

In this section an integration framework for buffer management and bandwidth scheduling for shared buffer switches is devised. High-speed Internet requires support for priority traffic such as video conferencing and VoIP. Support for best-effort traffic is not sufficient. The power of shared buffer switches can be enhanced by providing support for delay and loss sensitive network traffic.

In this framework the shared memory is shared among output ports and it is further shared among different number of traffic classes (see Fig. 5.4). Each traffic class is a FIFO queue which is provided with a portion of finite output port bandwidth and buffer space from the shared finite buffer pool. Each sub-queue claims buffer space from the shared memory pool, which is proportional to arrival rate to that traffic class

(λ_{ij}) and inversely proportional to service rate of the traffic class (μ_{ij}). Nevertheless, the sum of the length of all sub-queues must not exceed the buffer size:

$$M \geq Q(t) = \sum_{i=1}^{N} \sum_{j=1}^{n} q_{ij}(t), \tag{5.26}$$

Where q_{ij} denotes the queue length in terms of number of packets in traffic class j of output port i. Each port has a dedicated sorter that classifies packets according to the priority field in the packet header information; and a scheduler that allocates bandwidth using a scheduling policy such as weighted round robin (WRR), weighted fair queuing (WFQ) etc. (Cisco Systems 2000). As in the best-effort traffic, one or more sub-queues might be growing in length and thus dominating the shared buffer pool. This is due to high volume of arrivals as in best-effort traffic case and/or low service rates (i.e. low bandwidth allocated by the scheduler). A dynamic threshold scheme can be used to avoid saturation of buffers by an overloaded sub-queue. Decay function threshold scheme is a per-queue threshold scheme that controls a queue based on its overall buffer occupancy with respect to unoccupied buffer space. By this way, buffer management and bandwidth allocation can be employed in the same framework. Hence the threshold equation can be written as:

$$T_{ij}(t) = \frac{M}{c^{\left\lfloor \frac{q_{ij}(t)}{B+1} \right\rfloor}}. \tag{5.27}$$

The scheduling mechanism decides which queue to serve next by using a bandwidth allocation policy. Bandwidth allocation at each output port policy may assign

different weightings for different classes of traffic. Moreover, each port may employ totally different bandwidth allocation scheme.

When employing the buffer management, if a global threshold is to be used, threshold scheme is unable to discriminate between high and low priority packets. On arrival each packet is treated in the same way and packets are admitted or rejected regardless of their priority. On the other hand, per-queue threshold works in a fair fashion. Packet admission and rejection depends on the current status of each queue. Per-queue threshold determines admission and rejection of packets according to the length of each sub-queue at any given time. For instance, high priority traffic receives more bandwidth and therefore their queue lengths are shorter and low priority queues grow in length as they are served less frequently. At this stage per-queue threshold prevents low priority queue from growing further by rejecting excess arrivals.

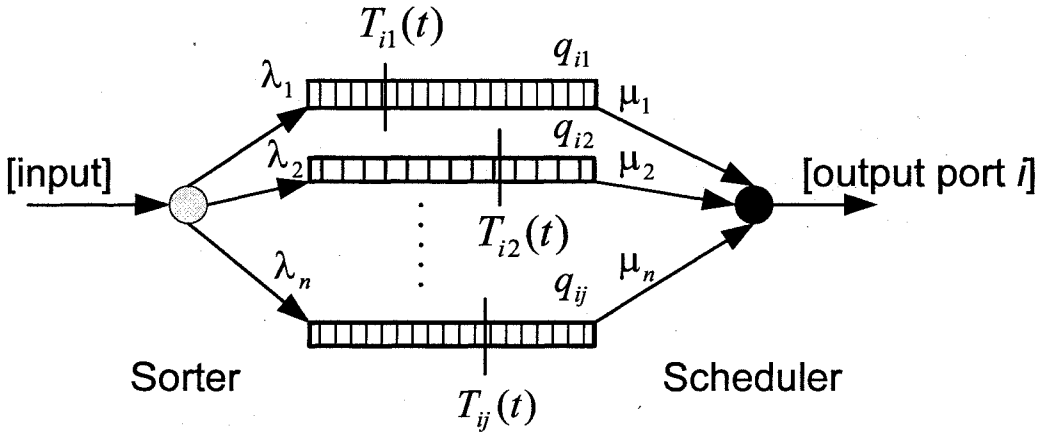


Fig. 5.4 – Integrated buffer management and scheduling

In this framework, buffer allocation policy is also responsible from sharing buffer space among different priority classes whilst scheduler is responsible from allocating

bandwidth for different traffic streams. These modules are loosely coupled to reduce the communication overhead between two management schemes. Per-queue threshold scheme is suitable for this framework, because each priority queue (sub-queue) is treated individually during packet admission and bandwidth allocation.

5.6. Summary

In this chapter a new dynamic buffer management scheme for shared buffer switch architectures has been proposed. Firstly, the main features of an optimal policy were discussed. Secondly, the decay function threshold policy was derived by using a systematic approach. In this section, two versions of the decay function threshold policy were derived. Suggested threshold policy tries to improve the performance of existing policies by monitoring more than one environment variable such as current occupancy of an output port queue and unoccupied buffer space. Thirdly, an analysis was carried out to observe how active queues converge to a steady state allocation. It was shown that active queues reject their arrivals in order to release some buffer space for newly active queues and queue lengths are re-stabilised by the threshold policy. Finally an integration framework for buffer management and bandwidth scheduling was proposed for shared buffer switch architectures in order to provide support for QoS traffic.

The next chapter presents the model devised for simulating the shared buffer switch architecture. The aim of developing this simulation program is to demonstrate the performance characteristics of dynamic buffer management schemes under real traffic conditions.

CHAPTER 6

SIMULATION MODEL AND IMPLEMENTATION

6.1. Introduction

This chapter provides a detailed discussion on simulation program developed to measure and observe performance characteristics of buffer management policies used in the shared memory packet switch architecture. The purpose of modelling and developing such simulation program is to have a comprehensive understanding of the switch architecture as well as the factors, such as traffic source models, affecting this system. Moreover, it provides flexibility when implementing traffic source models and buffer management policies.

The next section provides an overview of the Parallel Virtual Machine framework. Developed simulation program is based on this platform. Section 6.3 describes the system model. This model consists of the main components that establish the whole system. Section 6.4 provides a short overview of the traffic source models. These models help to emulate packet flows in a real network environment. Performance metrics considered in simulations are presented in Section 6.5. Implementation details of the simulation program are provided in Section 6.6. Finally, a short discussion on main issues such as duration of a single run, randomness, benchmarking, concerning this simulation model is provided in Section 6.7.

6.2. An Overview of the Parallel Virtual Machine

The parallel virtual machine (PVM) provides a framework within which parallel programs can be developed and executed in an efficient manner regardless of the underlying hardware (Geist et al. 1994). PVM allows a cluster of heterogonous computers to be used as single parallel machine. Message passing and job scheduling across the network are transparent to the parallel program and hence to the developer. Physically interconnected computers can be added and removed to/from the parallel virtual machine environment and programs can be coded as a set of tasks or routines. PVM enables programmers to use the PVM library functions for creating and killing tasks, sending and receiving data structures, task synchronisation, message broadcasting etc. Tasks can be initiated by other tasks and each task works as an independent asynchronous sequential thread. The unit of parallelism in PVM is task and many tasks may execute on a single processor. Communication between tasks is carried out via messages. Messages are exchanged between tasks through a duplex pipeline.

PVM supports C, C++ and Fortran programming languages and programs can be developed and executed under Linux operating system. Different versions of Linux provide support for the PVM environment. The general methodology for developing applications with PVM is as follows:

i. Sequential programs are coded in C, C++ or Fortran. These contain calls to PVM library functions.
ii. These programs are compiled on one of the host machines and resulting object files are placed on a host accessible by all machines in the host pool.

iii. In order to start the application, user executes the master task from one of the hosts and eventually other tasks are initiated by the master task (i.e. master-slave model).

iv. Tasks are identified by their unique task id (TID). By this way different tasks know each other and messages are exchanged by using TIDs.

v. Once a task is completed, the control is passed on to the master. When all tasks are completed, a master task either exits or spawns new tasks for further execution.

Several other parallel programming packages, such as Express (Flower 1994), MPI (Pacheco 1996), Linda System (Gelernter 1985), JPVM (Ferrari 1997), have been developed to enable parallel and distributed programming. However, there are a number of reasons for choosing this specific parallel environment for simulation purposes: (i) PVM is free, (ii) it is open source and widely used for research purposes, (iii) good documentation and sample codes, (iv) it supports programming with C/C++ programming language, (v) programming paradigm is straightforward, (vi) off-the-shelf heterogeneous computers can be used to create this parallel environment (this makes it cost-effective), and (vii) Unix/Linux operating system is reliable and has robust random number generators.

A parallel discrete event simulation program can be viewed as a collection of interacting sequential simulators. The main objectives of simulating a network node in a parallel platform include: (i) a realistic environment can be achieved where different parts of a network switch architecture, such as traffic sources, packet sinks, switch arbitrator, buffer management unit etc., work independently from each other in an

asynchronous manner, and (ii) simulation times can be reduced using the parallel processing paradigm. Reasonable results can only be obtained by running simulations over a longer period, thus requiring a much lengthy executions (Frost and Melamed 1994). To avoid this, more machines can be added to the PVM host pool to reduce the simulation times by increasing the processing power (Bhatt et al. 1998). The main dilemma with this approach is synchronising message communications between machines in the PVM host pool (Nicol and Liu 2002). The next section describes the system model used in the implementation of the *N*-by-*N* shared buffer switch architecture.

6.3. System Model

As described in Chapter four, a shared memory packet switch consists of *N* inputs/outputs and a RAM memory shared among *N* output ports. By using this simple description it is possible to identify the main parts required to model this system. Each part can be considered as an individual task. The types of tasks that form this system are (i) the switching module, (ii) traffic source and (iii) packet sink. The overall system model is depicted in the Fig. 6.1, and its main modules are described in the following sections.

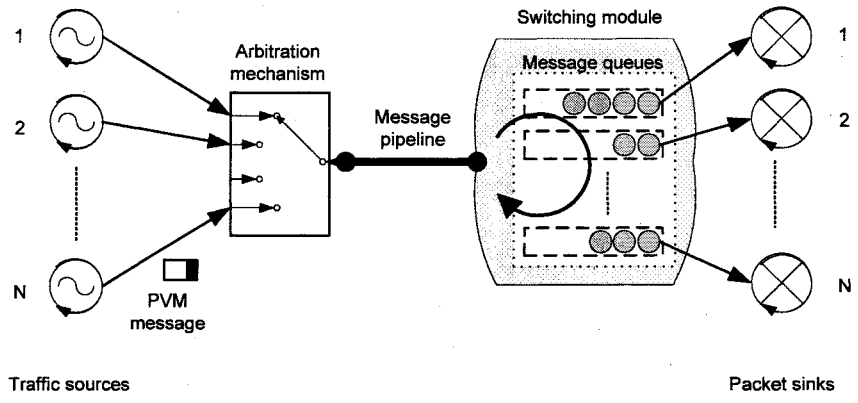


Fig. 6.1 – Shared buffer switch simulation model

6.3.1. Switching module

This module is responsible from admitting/rejecting packets, storing packets in a FIFO order at each output port, keeping a current state of the buffer memory, and employing buffer management and QoS policies. Arbitration of packet arrivals is handled by the PVM by default. In this model, a PVM message can be considered as a network packet/cell. Messages arriving at the switching unit are placed in the communication pipeline where they wait until they are processed by the switching module. Messages arrive in random order and it is very unlikely for two messages to arrive at the same instance. Even if messages arrive at the same time instance, PVM employs its arbitration mechanism to serialise messages in the message pipeline. As in a real switching system, switching module works N times the line speed in order to handle all arrivals as quick as possible. Otherwise, the number of packets in the pipeline grows rapidly forming a backlog of messages. As single network node is simulated, only one instance of this task is required.

6.3.2. Traffic source

This type of task is only responsible from generating packets according to a traffic source model, such as the Poisson, Markov Modulated Poisson Process (MMPP), interrupted Poisson process (IPP), On-off model etc., and feeding them into the switch. The traffic source models used in this simulation model will be explained in detail later in this chapter. A typical packet is composed of a header and a payload. Packet header contains information about packet destination and its priority level. The following data structure is used for representing a packet:

```
struct packet {
        long int source;
        long int destination;
        byte priority;
        struct packet_payload payload;
};
```

Source is the originating task and destination denotes the target task (i.e. output port, packet sink). Packet priority can be represented by using a single byte. For instance when two priority levels are required, low (0) and high (1) are used to represent the priority level of the packet. In multiple priority case, different classes can be represented in the form of bit settings (e.g. 2^8 possible priority classes). This data structure is wrapped into PVM message and exchanged between tasks. In an $N \times N$ switch, N number of instances of this task is required (i.e. one per switch input). Each instance works independently and traffic models and loads are the same for all traffic sources (i.e. replicate traffic sources).

6.3.3. Packet sink

This type of task is responsible from removing a packet from the head of line of the corresponding queue. Output ports of the switch architecture are connected to packets sinks. Packets sinks check the head of line of each queue after every time interval. This interval can be modelled as deterministic or by using a type of renewal model such as Poisson, Geometric etc. In simulations deterministic approach is preferred as it is widely considered in the literature and it makes easier to observe delay over simulation time caused by n-burst arrival process. After a certain fixed time period, each packet sink checks the head of the corresponding queue. If a packet exists in the

corresponding queue, then it is removed from the queue and the buffer state of the switch unit is updated. Otherwise no action is taken and after further time period the packet sink checks the queue. As in traffic sources, one packet sink is required for each output port.

All three types of tasks communicate with each other by using PVM's message passing feature. As in real world network nodes, modules work independent from each other in an asynchronous fashion. This approach helps us to imitate the real world in the laboratory environment. This simulation model is programmed as console application by using the C programming language. All three tasks are coded as three different files. The master task (switching module) activates the rest of the tasks according to the given switch size. Traffic sources generate predetermined number of packets according to a given traffic load. Once all packets are generated, traffic sources terminate. Similarly, packet sinks terminate once all packets are cleared from the switch module message queues. Simulation parameters such as switch size, buffer size, traffic load etc., are entered in the argument list when the application is run from the console.

6.4. Traffic Source Models and QoS

Performance simulation models require accurate traffic models that capture the statistical characteristics of actual traffic. If models fail to represent actual traffic accurately then one can easily underestimate or overestimate system performance under consideration.

A stream of requests can be characterised by successive observations, $...,X(t_{n-1}),X(t_n),X(t_{n+1}),...$ at time instants $...,t_{n-1},t_n,t_{n+1},....$ With these observations it is possible to describe inter-arrival times of packets/cells or packet sizes in variable packet size networks (e.g. IP networks). Normally, variable $X(t_i)$ is modelled by using a probability distribution function and time index *t*. If the set of values produced by the distribution function is finite, then it is called discrete-state process; otherwise continuous-state process (Frost and Melamed 1994).

6.4.1. Renewal traffic models

In a renewal model, $X(t_i)$ is independent identically distributed (i.i.d.). That is, observation (or arrival of a packet) at time *t* is independent of observations (arrivals) before or in the future. Therefore this type of models lacks autocorrelation between successive arrivals. A Poisson process is characterised as a renewal process whose inter-arrival times are exponentially distributed with mean arrival rate λ (Adas 1997). Hence, the *n*th inter-arrival time A_n is given by an exponential distribution given by (Adas 1997):

$$P\{A_n \leq \tau\} = 1 - e^{-\lambda\tau}. \tag{6.1}$$

The number of arrivals in duration τ is modelled using a counting process given by (Adas 1997):

$$P\{N(\tau) = n\} = \exp(-\lambda\tau)(\lambda\tau)^n / n!. \tag{6.2}$$

Bernoulli processes are another form of renewal process where arrivals can only take place at some time slot such that the probability of an arrival in a time slot is *p* and is

independent of others (Adas 1997). The number of arrivals in slot k is binomially distributed given by:

$$P\{N_k = n\} = \binom{k}{n} p^n (1-p)^{k-n}, \tag{6.3}$$

where $0 < n < k$. The number of time slots between two arrivals is geometrically distributed and is given as:

$$P\{A_n = j\} = p(1-p)^j, \tag{6.4}$$

where j is a positive integer.

6.4.2. Markov models

Dependencies between observations can be defined by Markov processes (Frost and Melamed 1994). A Markov process with discrete state space is called Markov chain, if the probability of the next state j depends on the current state i and this probability if given as p_{ij}. A Markov process exploits memoryless property (Frost and Melamed 1994). That is, once a particular state has been visited, the future behaviour is always the same regardless of how the state is reached. Hence dependency is one unit back in time and independent of the time spend in current state.

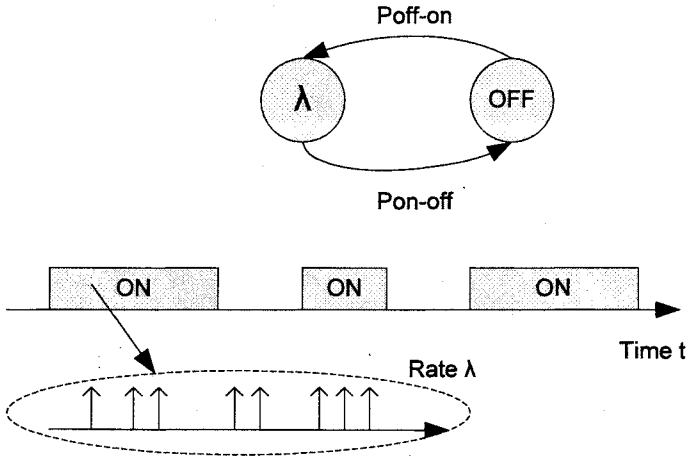


Fig. 6.2 – Interrupted Poisson process (IPP)

ON-OFF and interrupted Poisson process (IPP) traffic source models are two examples of Markov source models (Adas 1997). The ON-OFF source model describes the source model for voice. In this model packets arrive with fixed intervals during ON state and no packets are generated during OFF state. The time spent during ON and OFF durations (T_{ON} and T_{OFF}) is exponentially distributed with means $1/\alpha$ and $1/\beta$, respectively. The interrupted Poisson process is also a two-state process. Arrivals occur during ON state with rate λ according to a Poisson distribution (see Fig. 6.2). Thus the normalised offered load p is given as:

$$p = \frac{\lambda . T_{ON}}{T_{ON} + T_{OFF}} \tag{6.5}$$

Markov renewal process is a two-state process with states s_1 and s_2. This process alternates between two states without any self-transition. In one state, traffic phase is zero and in the other it is one. If the mean sojourn time for s_1 and s_2 are given as d_1 and d_2, then the steady state probabilities of being in these states are given as $d_1/(d_1 + d_2)$ and $d_2/(d_1 + d_2)$, respectively.

Markov modulated Poisson process (MMPP) is a doubly stochastic process that uses an auxiliary Markov process (Frost and Melamed 1994). This imposes current state of the Markov process to control the probability distribution of the traffic. In other words, in state s_k, the arrivals occur according to a Poisson process with rate λ_k. The MMPP model aims to model time-varying sources. Another example of Markov modulated traffic source model is Markov modulated fluid model (MMFM).

Other types of traffic models include regression models, such as autoregressive models, TES, and self-similar traffic models such as Fractional ARIMA, Fractional Brownian Motion (Adas 1997; Frost and Melamed 1994). Self-similar traffic models form an important part of traffic modelling. For instance Ethernet traffic is self-similar which exhibits structural similarities over a long-range of time scales (Adas 1997). Self-similar processes (also known as fractals) can be defined using heavy-tailed distributions (or long-tailed distributions). A heavy-tailed (or long-tailed) probability distribution is one that assigns relatively high probabilities to regions far from the mean (Frost and Melamed 1994). Examples of self-similar traffic include WWW, TCP, FTP, TELNET and VBR video. The simplest way of generating self-similar traffic is superimposing a large number of heavy-tailed ON-OFF processes. Durations of ON and OFF are derived from a heavy-tailed distribution such as Pareto distribution.

The interrupted Poisson process will be considered in simulations, as it offers a general notion for bursty traffic and it is easy to implement. In the simulation model, bursty traffic sources are super-positioned to the message pipeline (see Fig. 6.1). Although this does not provide self-similar traffic characteristics, combining N independent ON-OFF sources defines the N-burst process (Adas 1997).

6.4.3. Imbalanced traffic

In the uniform traffic case, each traffic burst has equal probabilities of being destined to any of the output ports. There are two different types of output ports in imbalanced input traffic case: (i) moderately loaded and (ii) heavily loaded. Special case of imbalanced traffic is called the hotspot traffic and in this case only one output port is

heavily loaded (i.e. hotspot port). In imbalanced traffic all input ports receive identical input traffic loads; however packets are addressed asymmetrically. The aim of using imbalanced input traffic is to create a situation where some ports have higher demand for buffer space due to higher volume of traffic on that port. In this way, it is possible to observe the performance implications of heavily loaded ports on the shared buffer memory.

6.4.4. Multicast traffic

When a packet arrives at one of the switch inputs, its replicates are produced and sent out to those output ports other than the original destination. Note that replicates are produced on arrival when the original packet is waiting to enter the switch buffers. Each replicate joins the end of the corresponding output port queue. Meanwhile buffer management policy decides whether to admit or block replicates at each queue. In such a case, replicates that are joining relatively smaller queues might be admitted to the switch buffers, whilst others might be blocked if the corresponding queue length is greater than the threshold. In the absence of a buffer management mechanism packets are admitted until the switch buffers become full.

Multicast probability determines the likelihood of multicasting a generated packet at a given time instant. If this probability is high, multicasting is very likely to take place; otherwise a situation close to unicast traffic is observed. Another parameter that defines the impact of multicast traffic is the multicast group size. Group size determines the number of packets to be reproduced once the multicasting decision is made. Group size is normally from one to N - 1, where N is the number of switch input/outputs. Group size equal to N - 1 is a special case, called broadcast. In this case

all output ports (apart from the original destination) receive a copy of the original packet.

6.4.5. Quality of Service traffic

The main purpose of the Quality of Service (QoS) traffic is to provide priority privileges across different traffic streams such as video and voice. QoS traffic offers dedicated bandwidth, controlled delay jitter and improved loss whilst ensuring fairness among various traffic classes. Different levels of QoS include best-effort service, differentiated service, and guaranteed service. Best-effort traffic is lack of QoS. All traffic streams are handled through a single FIFO queue. Differentiated services provide better service for preferred traffic streams. This is done by classification of traffic into separate FIFO queues and bandwidth allocation by employing schemes such as priority queueing (PQ), custom queueing (CQ), and weighted fair queueing (WFQ). Guaranteed service is an absolute reservation of network resources for specific network traffic (e.g. RSVP).

During simulations apart from best-effort traffic, differentiated services are also considered. In this case packets arriving at the output links are classified into two or more FIFO queues. This classification is carried out according to the priority field in the header of the packet (e.g. Type of Service bits in IPv4 and Traffic Class in IPv6). The traffic classes considered during simulations are high-priority (Class 1) and low-priority (Class 2). Each traffic class is allocated with a portion of available bandwidth, $c = w_i \Big/ \sum_j w_j$, by using a work-conserving round-robin scheme called weighted round-robin (WRR). This bandwidth allocation scheme serves queues in a

circular fashion and each queue receives dedicated service duration. This approach is work-conserving discipline. For instance, if a certain queue is empty in the beginning of its turn or becomes empty before it completes its service time, WRR immediately jumps to the next queue.

6.5. Performance Metrics

The performance metrics considered are cell loss rate (CLR) and average cell delay. The term "cell" is used to refer to a fixed-length packet. Calculation of cell loss rate is straightforward. It is the number of cells blocked from entering the switch buffers over the number of cells generated by the traffic sources. For QoS traffic, it is simply given as:

$$CLR = \frac{CLR_1 + CLR_2}{n}, \tag{6.6}$$

Where n denotes the number of cells generated during simulation. In order to obtain high confidence in produced results, simulations are repeated until required confidence is achieved (repeated sampling) or a vast number of cells are generated during a single run. For instance in order to observe 10^{-6} loss rate, at least 1×10^{6} cells must be generated. This has an impact on the simulation times; nevertheless a reasonable statistical confidence is achieved. Loss performances can be calculated for different traffic loads and the number of heavily loaded ports.

Average cell delay is computed by using a different method. Sampling method is used to record samples at different time instances. This is done for each queue and it is

averaged at the end of the simulation. Normalised average delay is calculated as below to determine the average waiting time when a cell enters the switch buffers:

$$\overline{d} = \frac{\sum_{j=1}^{k} q_1(j) + \sum_{j=1}^{k} q_2(j) + \ldots + \sum_{j=1}^{k} q_N(j)}{k.N} \tag{6.7}$$

Where k denotes the number of samples taken during the simulation between t_{start} and t_{end} and $q_i(j)$ is the number of cells in queue i during sampling point j.

More accurate results can be achieved by increasing the sampling rate given as $k^{-1}.(t_{end} - t_{start})$. The average delay is calculated for different traffic loads and number of heavily loaded ports. It is also possible to observe individual queue lengths and empty buffer space during a portion of simulation time. This helps to analyse the transient behaviour of a buffer allocation policy during various transition points.

6.6. Discussions on Implementation

The simulation model presented in this chapter aims to measure performance characteristics of different buffer management algorithms. Note that there are number of points to consider during implementation of this simulation environment.

Firstly, a superior simulation program is based on a robust random number generator. Here, the developed simulation program is based on the Linux operating system random number generator. This random number generator gathers environmental noise from device drivers and other sources such as mouse movements into an entropy pool from which random numbers are created. Entropy pool relates the ability

of an operating system to produce random numbers. When the entropy pool is empty, no random numbers are returned. Linux operating system generates numbers with high degree of randomness as entropy pool is always with sufficient noise.

Longer simulations are needed to create rare events in a simulation environment. In designed simulation environment, the duration of a single run is determined by the number of packets planned to be generated. In order to achieve quality results, the number of generated packets must be at least 1×10^6. As one might expect this means longer simulation times. For the same number of generated packets, as the traffic load becomes smaller, simulation times increase. This is because inter-arrival times of packets become longer. Therefore, it is possible to conclude that the number of generated packets as well as the input load determines the overall duration of simulations.

Another factor that affects simulation time is the switch size. Simulation times increase exponentially with the switch size. This is because, more tasks are generated (i.e. traffic generators and packet sinks) and PVM environment partitions the available processing power among these tasks. Improved processing power results in reduced simulation times (e.g. one processor per task).

Benchmarking is important to determine the validity of produced results. Results produced by this simulation program can be compared with the implemented buffer management policies in other studies to iteratively refine and validate the developed simulation program (see Fig. 6.3). Besides simulating and contrasting buffer

management policies discussed in the literature survey by using the same platform is also considered as benchmarking for the policy that proposed in this study.

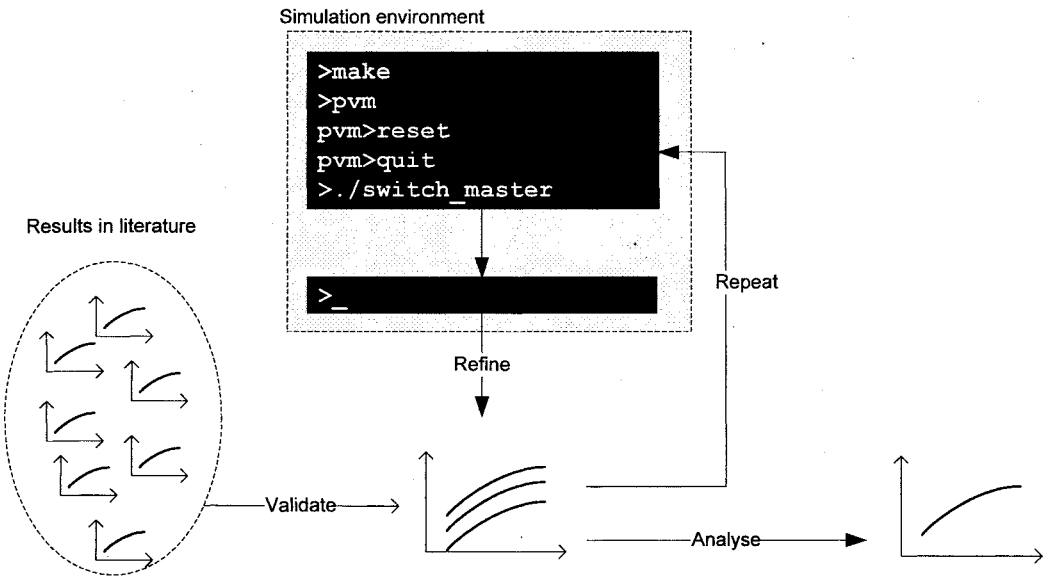


Fig. 6.3 – Benchmarking and validation

6.7. Implementation

Implementation model for $N \times N$ share buffer packet switch is illustrated in Fig. 6.4. It consists of three source files, 'switch_master.c', 'ssender.c' and 'sreceiver.c', which are complete realisations of switching module, traffic source and packet sink, respectively, and it is based on master-slave model (e.g. switching module being the master process). Each source file is coded as a sequential program. A 'make' file is used to create executables for each source file. More than one process (task) can be created, which are reproduced from the same executable. However, this excludes the

master process as all child processes are created and replicated by this process. The execution of a typical simulation is as follows:

1. Make file is used to create three executables: master (switching model), sender (traffic source) and receiver (packet sink)
2. PVM is started and reset
3. Master program 'switch_master.exe' is executed which in turn creates desired number of traffic source and packet sink child processes
4. Master program sends an initiation message to all child processes to start and synchronise them. After this stage, each process works sequentially, but uses an equal portion of processing power. By this way, a notion of parallelism is achieved even on a single processor environment
5. When all child processes finish their job, control is then returned back to the master process. Master process gathers results from child processes, computes cell loss rate and average cell delay performances, writes results to a text file and returns control back to the operating system

Parameters, such as buffer size, switch size, traffic load etc., are hard coded in the source files. For instance, when a different traffic load is selected, executables must be created again. Traffic source model can be easily changed by simply replacing the 'ssender.c' source file with another implemented traffic model. In the same way, other cell departure models can be employed by replacing 'sreceiver.c' source file. Buffer management policies discussed in Chapter four and proposed policy are implemented in the master file.

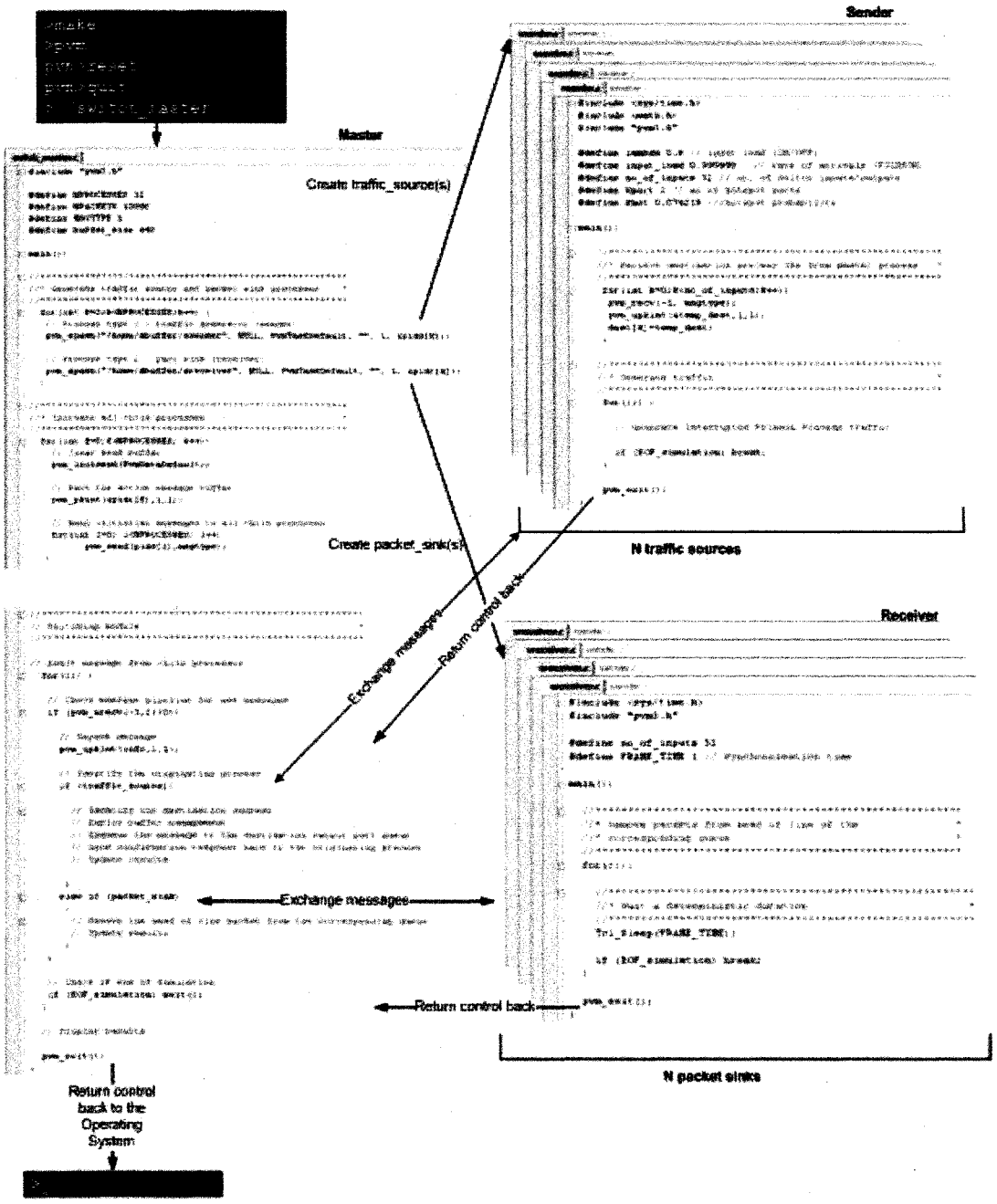


Fig. 6.4 – Implementation model

6.8. Summary

In this chapter, the model for simulating shared buffer switch architecture has been presented. This model consists of a switching module, traffic sources and packet

sinks. Traffic sources can be implemented by using any traffic source model described in the literature. Packet sinks remove packets from head of line of queues using specified departure models. Switching module unit implements the buffer management policy and keeps the current state of the shared buffer. The aim of developing such simulation program is to achieve realistic network environment so that performance characteristics of the shared buffer switch architecture could be measured and observed under certain traffic conditions.

The next chapter presents the results obtained by simulating various buffer management policies discussed in the literature as well as the policy proposed in this study. Simulation results are obtained by using the program modelled and developed in this chapter.

CHAPTER 7

SIMULATION RESULTS AND ANALYSIS

7.1. Introduction

In this chapter, performance characteristics of dynamic threshold policies, namely dynamic thresholds (DT), dynamic queue threshold (DQT), partial sharing partial partitioning (PSPP) and decay function threshold (DFT), are investigated and evaluated by means of simulations. Simulations are based on the assumptions made in Chapter six. Performance analysis will be conducted by employing three forms of network traffic: best-effort, multicast, and QoS traffic. First of all optimal control parameters for these policies will be determined by using the best-effort traffic. These policies will then be compared on the basis of chosen control parameters. One variant of DFT policy, DFT2, will further be investigated through comparative evaluation by using multicast and QoS traffic models. Benchmarking results that contrast dynamic threshold policy implemented in (Choudhury and Hahne 1997) and dynamic threshold policy implemented in this study are provided in Appendix B on the basis of similar fair share buffer size per output port and traffic conditions.

7.2. Best-Effort Traffic

In this section, a comparative analysis of dynamic threshold buffer management policies is conducted by using the best effort traffic model. The purpose of this conduct is to determine an optimal control parameter for each dynamic threshold policy, and contrast cell loss rate (CLR) and average delay performances. In this

context, the term "cell" will be used throughout this chapter to refer to a fixed-length packet.

7.2.1. Optimising control parameters for dynamic threshold policies

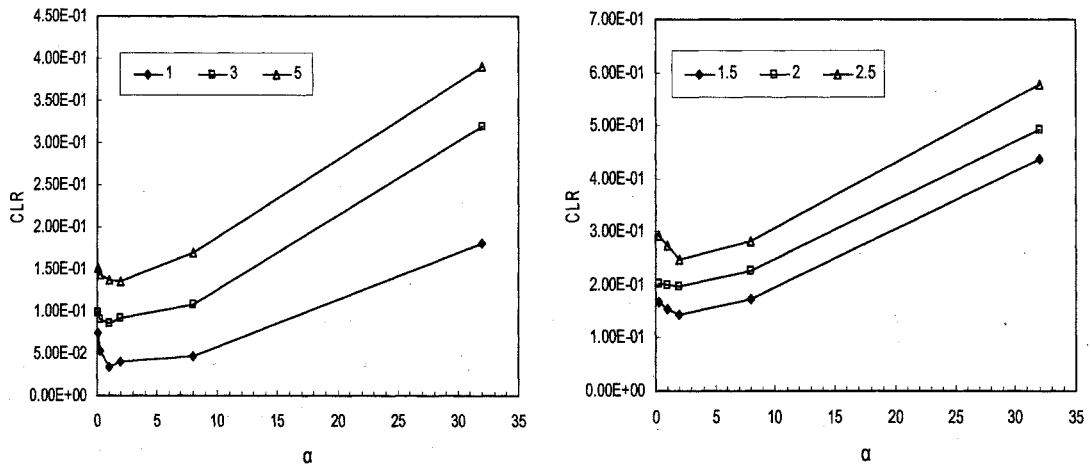
In order to determine optimal control parameters for dynamic threshold policies, two particular cases have been taken into account. In the first case, different number of heavily loaded ports, $h = \{1, 3, 5\}$, is considered while constraining the load on heavily loaded ports as $L_p = 0.95$. In the latter, the number of heavily loaded ports is kept constant as $h = 3$ and loads on heavily loaded ports are varied, $L_p = \{1.5, 2.0, 2.5\}$. Parameters for this simulation are given in Table 7.1. Switch size is chosen as $N = 32$ to make simulation manageable (see Section 6.6. for discussion on implementation) and buffer size of 640 cells is chosen to provide a nominal configuration with a fair share size of 20 cells per output port. Mean burst length is given as 100 frames where packets arrive with 100% rate.

Table 7.1 – Simulation parameters for optimising control parameters for dynamic threshold policies

Parameter	**Value**
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Case 1	
Number of heavily loaded ports (h)	{1, 3, 5}
Load on heavily loaded ports (L_p)	0.95
Case 2	
Number of heavily loaded ports (h)	3
Load on heavily loaded ports (L_p)	{1.5, 2.0, 2.5}

Figure 7.1 shows cell loss rate (CLR) against the DT factor α for different values of heavily loaded ports and load on heavily loaded ports. Control parameter α is varied

from 2^{-2} to 2^{5}. For both cases the loss rate increases with α for all values of heavily loaded ports and load on heavily loaded ports. In Fig. 7.1(a), low loss performance is achieved for α = 1.0. In case of very large values of α, threshold has no control over misbehaving queues, which is somewhat equivalent to complete sharing. For very small values of α, threshold policy is very restrictive over misbehaving queues which results in excessive rejection of cells. It is possible to observe that when the number of heavily loaded ports *h* is equal to 5, optimal α value shifts to 2^{1}. This raises concerns about the robustness of this policy. The effect of load on heavily loaded ports is best seen in Fig. 7.1(b), where the optimal value is 2^{1}. Based on these results one could state that α is optimal from 2^{-1} to 2^{1}, and hence its value is chosen to be one for comparative study.

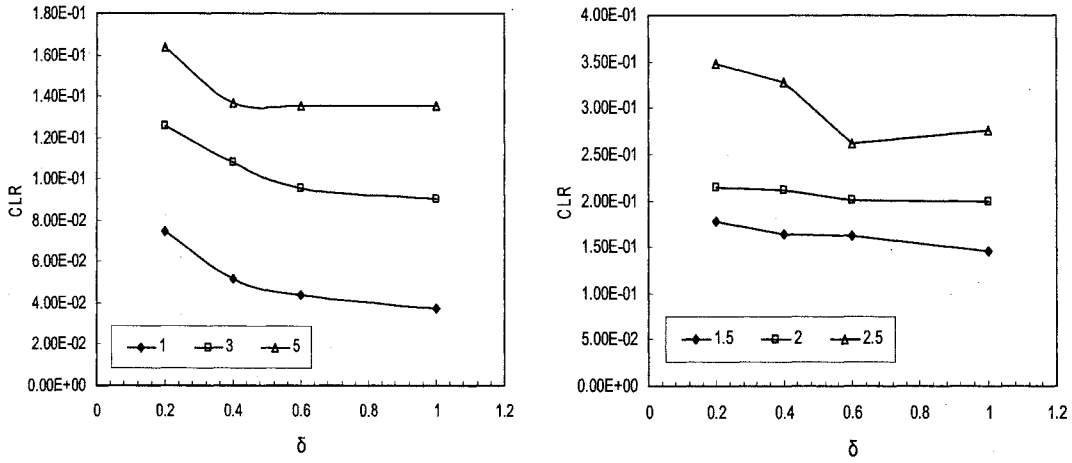


(a) (b)

Fig. 7.1 – Cell loss rate versus the DT factor α, (a) for different values of heavily loaded ports *h* and (b) load on heavily loaded ports L_p

CLR performance for a range of *h* and L_p for DQT factor of δ is illustrated in Fig. 7.2 for DQT policy. CLR decreases with *h* and L_p due to imbalance and burstiness of input traffic respectively. For a particular case, i.e. $L_p = 2$, the optimum δ is at 0.6. For

very small values of δ value, DQT policy is rather restrictive, thus resulting in complete partitioning of the buffer memory. In most cases the optimal value of δ can be observed as one, hence this value is chosen for DQT policy for comparative study.



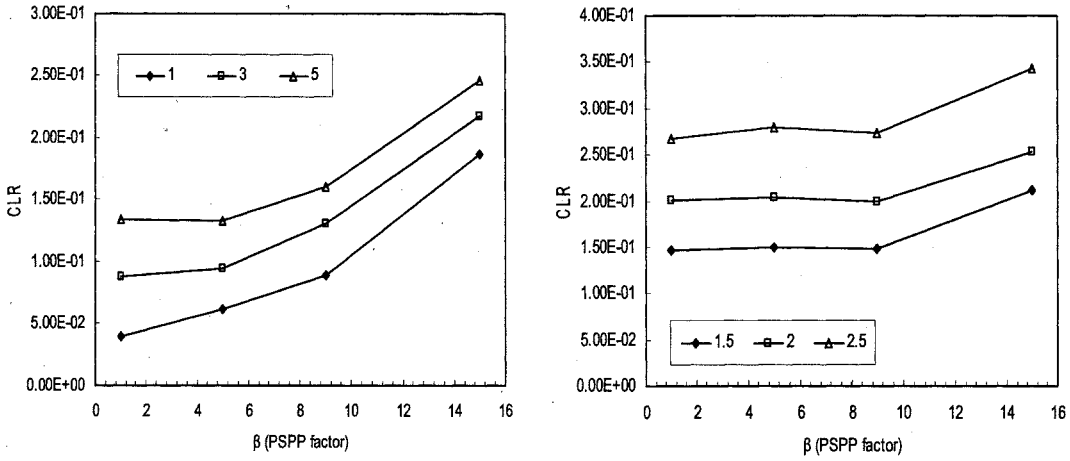
(a) (b)

Fig. 7.2 – Cell loss rate versus the DQT factor δ, (a) for different values of heavily loaded ports h and (b) load on heavily loaded ports L_p

Finally, CLR performance for a range of of h and L_p for PSPP factor β is shown in Fig. 7.3. It is possible to observe that CLR increases together with the PSPP factor value. When the PSPP factor β value is high, PSPP policy has a relaxed threshold control. Likewise control threshold is restrictive for smaller PSPP factor values. Comprehensive studies in (Chu et al. 2002) have shown that optimal PSPP factor value is from 1.5 to 2.0. Results in Fig. 7.3 are analogous. For convenience, this value is chosen as 1.5 for comparative study.

Control parameter values chosen for all three policies, namely DT, DQT and PSPP, are optimised to provide low CLR for the traffic source model assumptions made in this study (i.e. imbalanced input traffic with interrupted Poisson process source

model). Therefore, other traffic models, such as superimposed heavy tailed on-off, might prove otherwise.



(a) (b)

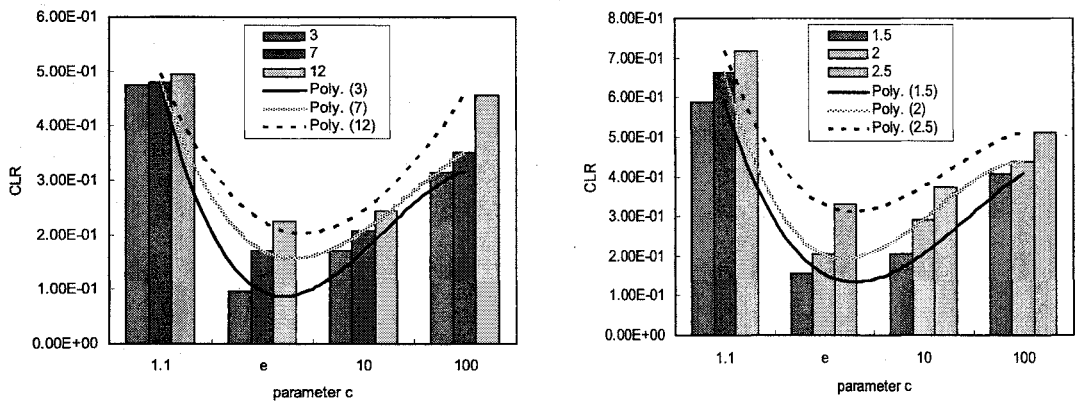
Fig. 7.3 – Cell loss rate versus the PSPP factor β, (a) for different values of heavily loaded ports h and (b) load on heavily loaded ports L_p

7.2.2. Tuning the share parameter for DFT

A series of simulations is carried to determine an optimal value for parameter c for both variants of the DFT policy. Parameters for this simulation setup are given in Table 7.2.

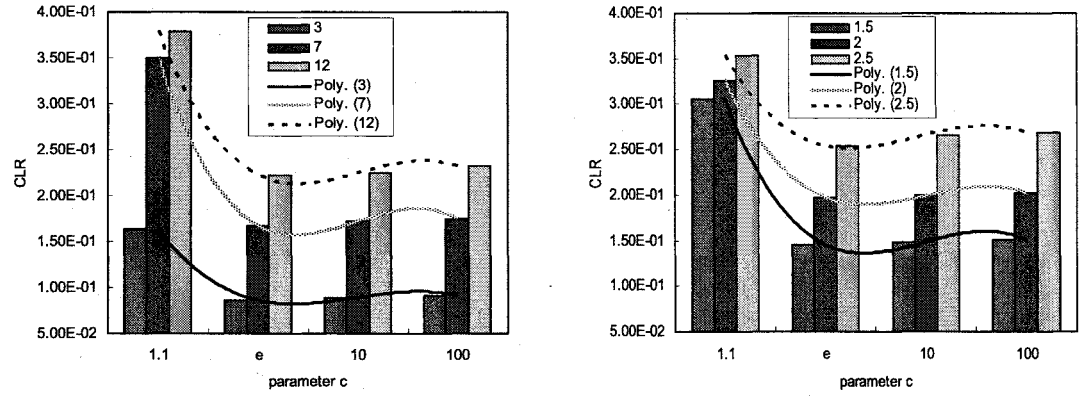
Table 7.2 – Simulation parameters for tuning share parameter for DFT policy

Parameter	**Value**
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Share parameter (c)	{1.1, e, 10, 100}
Case 1	
Number of heavily loaded ports (h)	{3, 7, 12}
Load on heavily loaded ports (L_p)	0.95
Case 2	
Number of heavily loaded ports (h)	3
Load on heavily loaded ports (L_p)	{1.5, 2.0, 2.5}



(a) (b)

Fig. 7.4 – Cell loss rate versus parameter c for DFT1, (a) for different values of heavily loaded ports h and (b) load on heavily loaded ports L_p



(a) (b)

Fig. 7.5 – Cell loss rate versus parameter c for DFT2, (a) for different values of heavily loaded ports h and (b) load on heavily loaded ports L_p

Two different situations have been considered to observe the behaviour of parameter c. Firstly, cell loss rate under different number of heavily loaded ports $h = \{3, 7, 12\}$, where load on each heavily loaded port is $L_p = 0.95$, see Figs. 7.4(a) and 7.5(a). Secondly, cell loss rate under various loads on heavily loaded ports $L_p = \{1.5, 2.0, 2.5\}$, where the number of heavily loaded ports is fixed as $h = 3$, see Figs. 7.4(b) and

7.5(b). Both of these cases are tested by using various values for c = {1.1, e, 10, 100}, where e = 2.71828183.

Table 7.3 – Average delay for DFT2 policy using different parameter c values under various traffic conditions

	No. of heavily loaded ports			Load on heavily loaded ports		
parameter c	**3**	**7**	**12**	**1.5**	**2.0**	**2.5**
1.1	19.65	19.77	19.71	19.66	19.66	19.66
e	17.70	19.01	19.27	17.49	18.09	17.65
10	17.20	17.88	18.54	14.83	16.42	15.80
100	15.01	16.29	17.37	12.79	14.78	13.43

Figure 7.4 illustrates the CLR performance versus parameter c for DFT1 policy for different values of heavily loaded ports and load on heavily loaded ports. In Fig. 7.4, it is possible to see that CLR performance is lowest for DFT1 policy when c = e. A similar setup is shown in Fig. 7.5 for DFT2 policy. In this case, there is a marginal difference for those values of c when c > e. When c is relatively a large value, control on queue lengths are rather strict, hence average queue length becomes smaller and more buffer space is left aside. To demonstrate this fact, average queue lengths under different traffic conditions are illustrated in Table 7.3 for DFT2 policy. These results are shown with six decimal points as originally produced by the simulation program. It is evident that when c is relatively large, thresholds become stricter resulting in poor buffer utilisation. Considering various traffic conditions and arrival patterns, optimisation of parameter c is extremely tedious and requires repetitive trials for optimisation purpose. One possible approach is using online optimisation algorithms, where parameter c is optimised given the lower and upper bounds for this parameter during real-time operation. This approach does not guarantee a good solution if the network traffic is exceptionally dynamic. Nevertheless, both variants of DFT have finer threshold granularities when compared with other policies as parameter c is of

floating point type. Moreover DFT2 employs two environment variables when computing the threshold value. Nevertheless, results produced in this simulation study provide estimate parameter c values for the proposed DFT policies.

Based on these observations, it is suggested that parameter c value is set as e and therefore this value will be used in the rest of this chapter. Nevertheless finer optimisations might be required to achieve desired performances.

7.2.3. Comparison of dynamic threshold policies

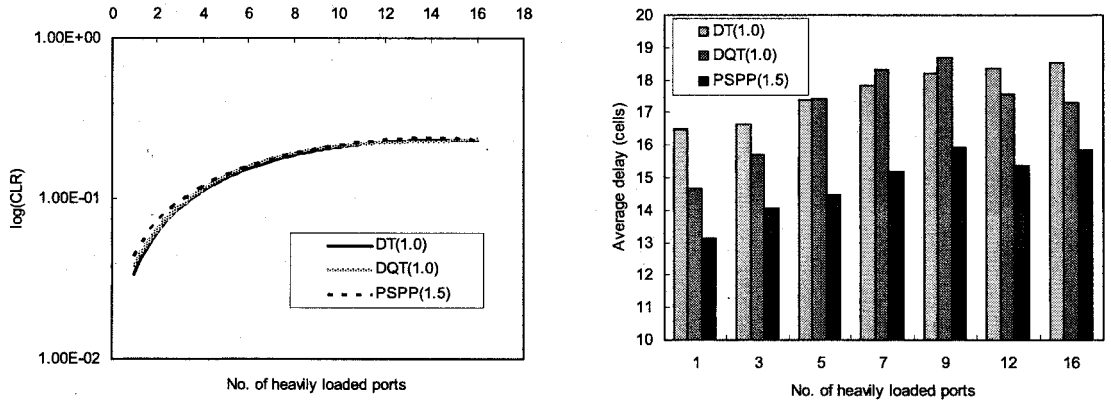
Performances of three well-known dynamic threshold policies, DT, DQT and PSPP, are compared in this section. Control parameters determined in Section 7.2.1 are adopted in the study and simulation parameters are given Table 7.4.

Table 7.4 – Simulation parameters for comparison of dynamic threshold policies

Parameter	Value
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Load on heavily loaded ports (L_p)	0.95

Figure 7.6 illustrates the effect of the number of heavily loaded ports h on the CLR and average delay performances for DT, DQT and PSPP. In Fig. 7.6(a), CLR axis is in logarithmic scale in order to smooth the lines. When the number of heavily loaded ports is high, there is no much difference between these policies. The reason for this is that buffers are saturated due to high volume of traffic and none of the buffer management policies have any impact to stabilise the performance of the switch. When the traffic is not very bursty, i.e. when the number of heavily loaded is h is

equal to one, it is possible to observe marginal differences. DT policy outperforms both DQT and PSPP policies in terms of CLR whilst PSPP performs the worst. In Fig. 7.6(b), DT offers relatively stable average cell delay. However, average delay for PSPP and DQT rises first and then drops when the traffic becomes bursty (e.g. number of heavily loaded ports greater than 9). PSPP scheme offers very low delay. The reason for this is that PSPP leaves more buffer space aside for lightly loaded ports. The threshold adopted by the PSPP is rather strict compared with DT and DQT. For this reason, more cells are dropped from the tail of heavily loaded ports. Unlike PSPP, DT is more tolerant to misbehaving queues allowing them to reach a certain degree of length. Amount of buffer space spared by DQT for moderately and lightly loaded output ports is somewhat between PSPP and DT which reflects to the CLR performance respectively.

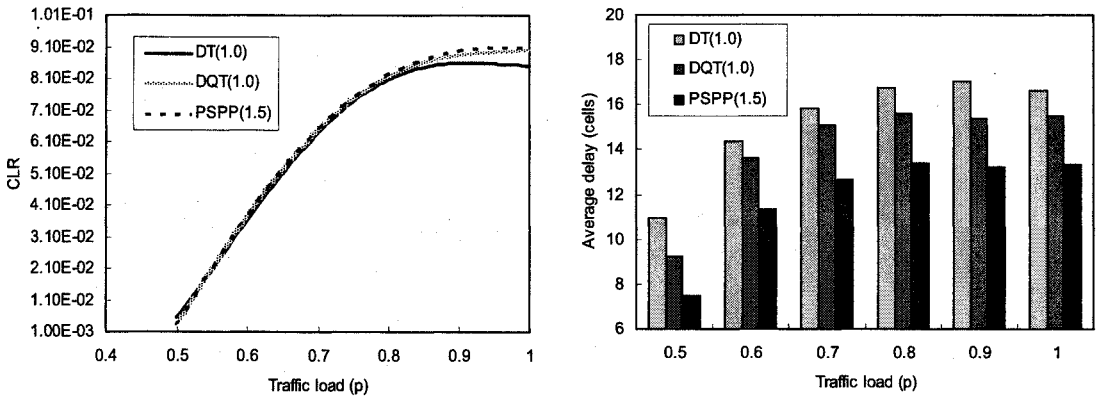


(a) (b)

Fig. 7.6 – (a) Cell loss rate performance and (b) average delay versus the number of heavily loaded ports for DT, DQT and PSPP

The effect of the traffic load on the CLR for DT, DQT and PSPP is shown in Fig. 7.7. In this case, the number of heavily loaded ports and the load on heavily loaded ports

are given as h = 3 and L_p = 0.95, respectively. Traffic load p is determined by adjusting arrival rate λ during T_{ON} duration and the duration of T_{OFF} (see Section 6.4.2). In Fig. 7.7(a), when the traffic load is light, all three policies offer similar loss performances. When the traffic load becomes heavy, the DT policy imposes better CLR performance. Both DQT and PSPP exploit similar CLR performances under various traffic loads. In Fig. 7.7(b), the average delay experienced for different traffic loads is shown. The PSPP policy offers the least delay regardless of the traffic load. It is possible to justify that there is a similar pattern among three policies as the traffic load increases. One might say that choosing the PSPP policy is better when the traffic load is light, as it provides the least delay for the same packet loss probability. In case of heavy loads, the DT policy offers the best loss rate with the trade-off of more delay. As in the previous results, DQT is compromise between DT and PSPP policies.



(a) (b)

Fig. 7.7 – (a) Cell loss rate performance, and (b) average delay, versus the traffic load for DT, DQT and PSPP

Finally, transient plot of queue lengths (in terms of number of cells) versus the simulation time for DT (α = 1.0), DT (α = 1.5), DQT (δ = 1.0), and PSPP (β = 1.5) is

depicted in Fig. 7.8. Simulation parameters used in this setup are given in Table 7.5. In Figs. 7.8(a) and 7.8(b) the transient plot for DT policy for $\alpha = 1.0$ and $\alpha = 1.5$ are shown. When $\alpha = 1.0$, the available buffer space is equal to the threshold; that is, buffer space spared for moderately and lightly loaded output ports at time t is equivalent to the threshold value at the same instance. When $\alpha = 1.5$, it is possible to see that the spared buffer space is less than the threshold value. Note that, queue length of an active queue, e.g. Q1, does not exceed the specified threshold value at any instance. It is also possible to notice that threshold is reactive and changing with every possible sampling point. In Fig. 7.8(c), the transient plot for DQT is shown for $\delta = 1.0$. With this control parameter value, DQT policy spares more buffer space than the DT policy with $\alpha = 1.0$ and $\alpha = 1.5$. This is only because threshold is relatively more restrictive. Threshold value for DQT is quite persistent throughout various sampling points despite the frequent changes in unused buffer space and the length of heavily loaded queue Q1. Lastly, results for PSPP policy is shown in Fig. 7.8(d). It is apparent that PSPP policy with $\beta = 1.5$ employs extremely restrictive threshold which results in increased unused buffer space set aside for inactive queues. Threshold function of PSPP is more reactive than DQT policy, where threshold value does not persist over long periods and slight changes in traffic conditions can be captured effectively. However, PSPP is less dynamic than DT policy. In other words, the difference between threshold upper and lower bounds over a sustained period of time is small. Such threshold policy is unable to realise long term buffer requirements of individual output ports.

Table 7.5 – Simulation parameters for transient plot of queue lengths

Parameter	Value
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Number of heavily loaded ports (h)	3
Load on heavily loaded ports (L_p)	0.95

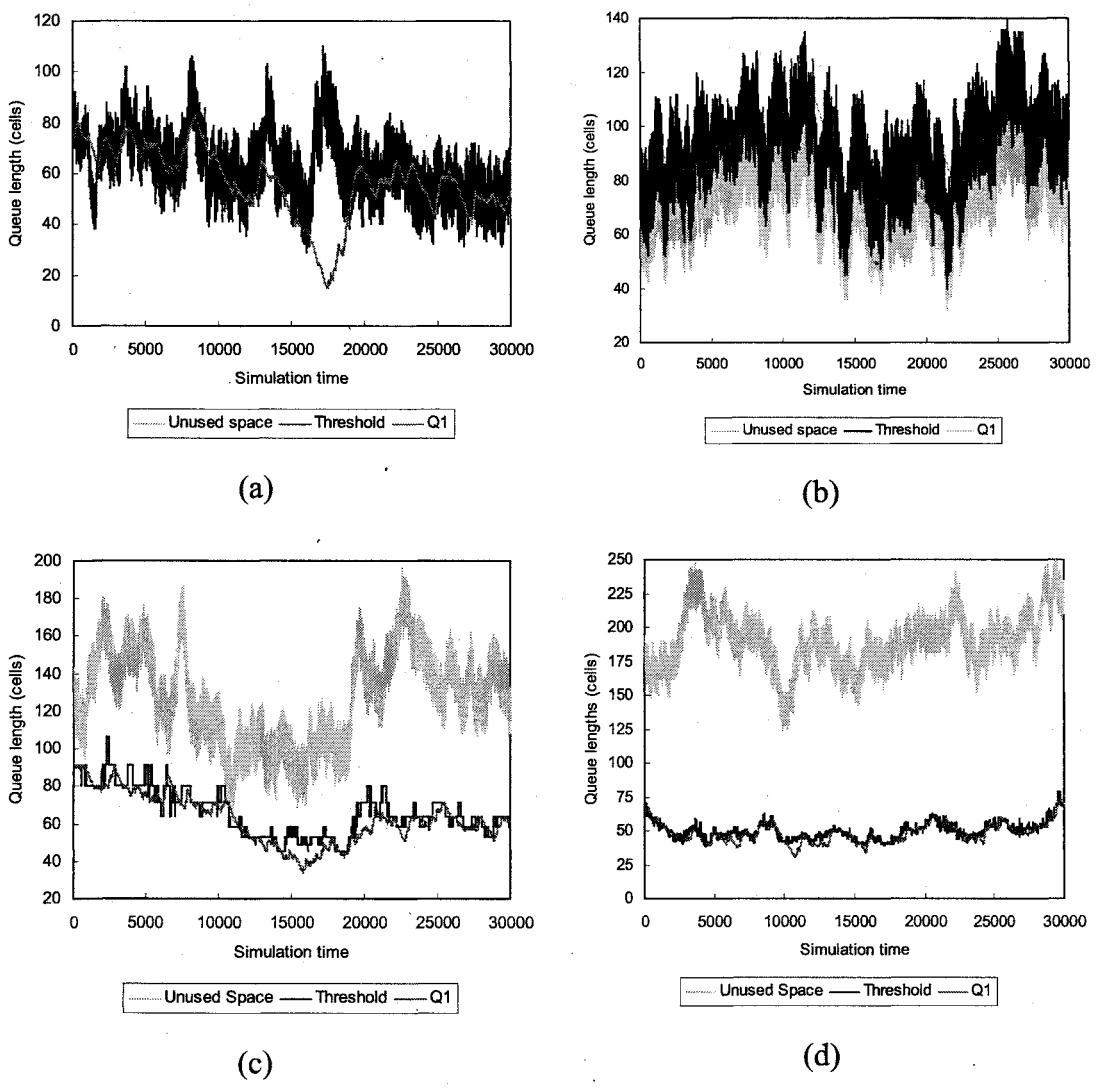


Fig. 7.8 – Queue lengths versus the simulation time: (a) DT ($\alpha = 1.0$), (b) DT ($\alpha = 1.5$), (c) DQT ($\delta = 1.0$), and (d) PSPP ($\beta = 1.5$)

The aim of sparing buffer space is to allow smaller queues to grow in length so that they can acquire some buffer space when the traffic intensity increases on these queues. Buffer space which is not in use by other output ports must be made available to active queues as the traffic intensity on these queues drives the buffer space requirements. For instance, the DT policy decreases the threshold when unused buffer space becomes smaller. This means a number of queues have become active and started acquiring buffer space. Therefore, extra buffer space used by already active queues must be returned back to newly active queues. As in DQT and PSPP policies, sparing more buffer space than actually required may result in poor utilisation of shared buffers.

7.2.4. Decay function threshold policy versus dynamic thresholds

Two variants of DFT policy, DFT1 and DFT2, and DT policy are evaluated via comparative study. Simulation parameters switch size and buffer size are given as N = 32 and M = 640 cells, respectively.

In Fig. 7.9, cell loss rate performances of DFT1, DFT2 and DT policies are compared for various loads on heavily loaded ports, L_p = {0.95, 1.10, 1.30}; traffic load is given as p = 0.8. As the load on heavily loaded ports increase, the difference between all three policies is indifferent. DFT2 and DT policies achieve similar CLR performances regardless of the number of heavily loaded ports and load on heavily loaded ports. DFT1 policy fails to match the performances of both DFT2 and DT policies when the traffic load is moderate (e.g. moderate load on active queues).

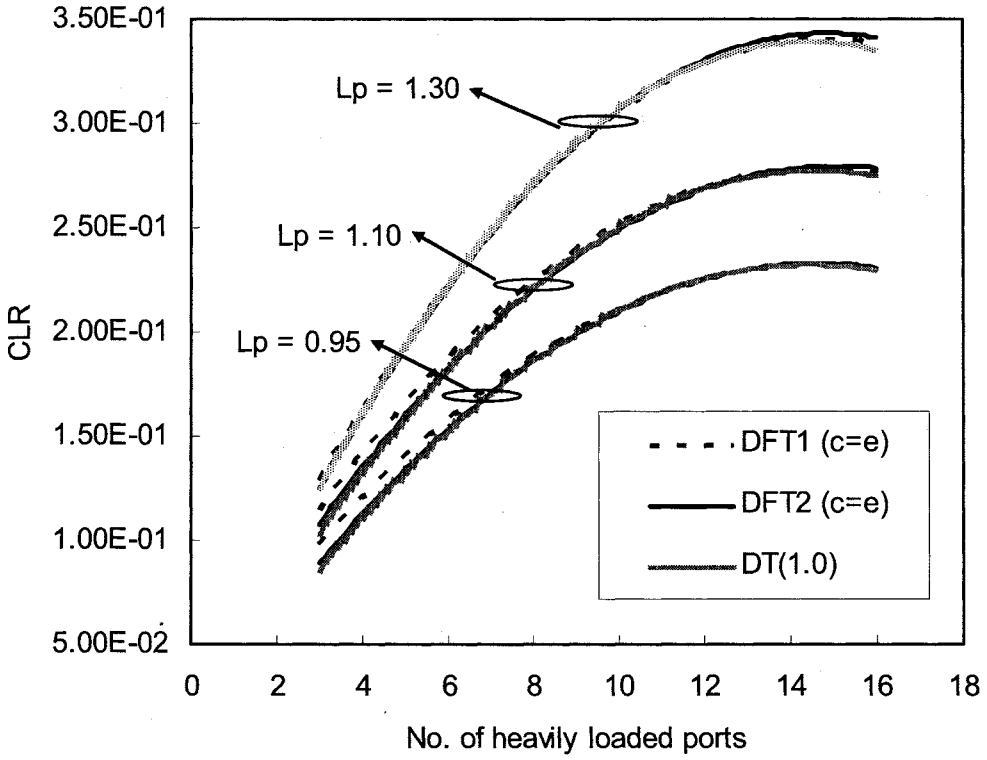


Fig. 7.9 – Cell loss rate performance versus the number of heavily loaded ports for various loads on heavily loaded ports ($L_p = 0.95$, $L_p = 1.10$, and $L_p = 1.30$)

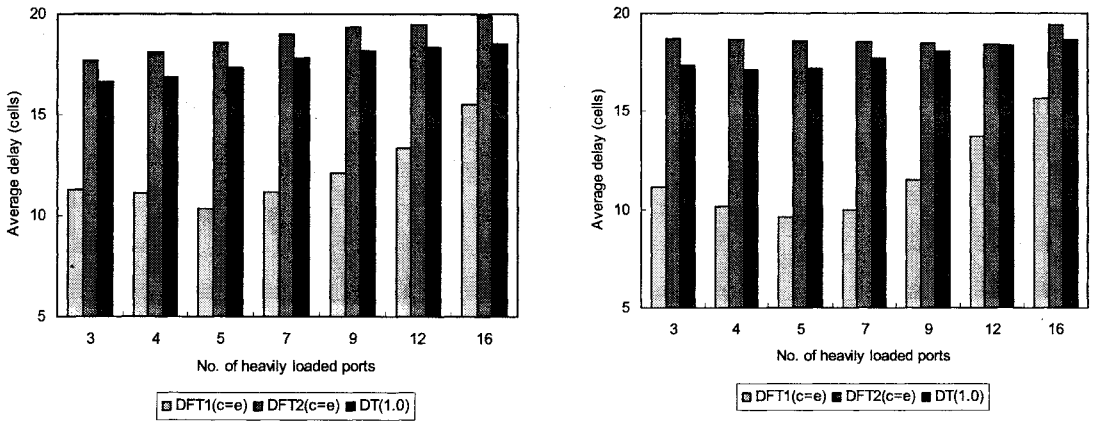
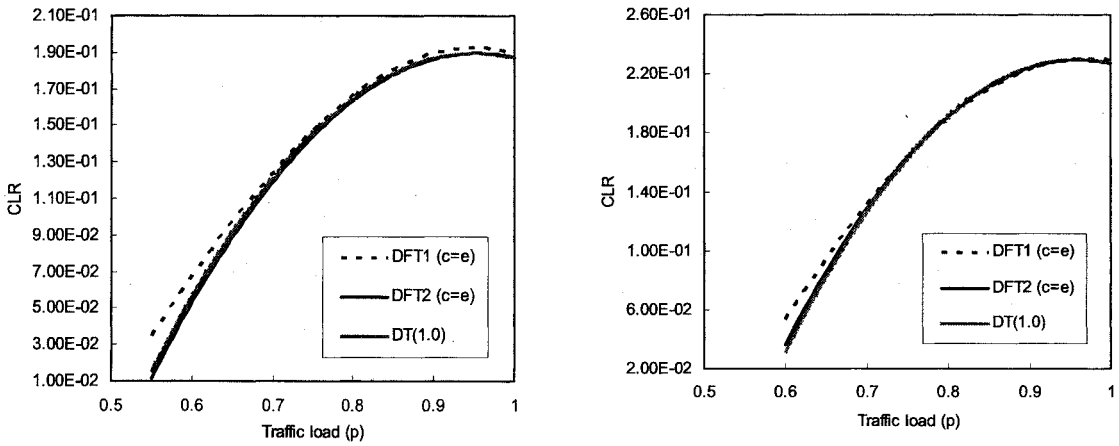


Fig. 7.10 –Average delay versus the number of heavily loaded ports for DFT1, DFT2 and DT policies: (a) $L_p = 0.95$, and (b) $L_p = 1.10$

Although the CLR performances of all three policies are alike, it is possible to observe the differences in the delay introduced by each policy for two different loads on heavily loaded ports, L_p = 0.95 and 1.10. This is illustrated in Fig. 7.10. DFT1 is strict on queue lengths; hence delay introduced by this policy is small. DFT2 makes more use of the shared buffer space. At some point, average delay is as long as fair share size allocated per queue, i.e. M/N. The DT policy offers average delay slightly less than DFT2. In normal circumstances, higher the delay the policy makes more use of the available buffer space. For instance, if buffer management policy is restrictive, queue lengths are smaller as cells are rejected from entering switch buffers. For this reason, DFT2 and DT policies admit more cells than DFT1 policy resulting in low loss. Nevertheless, switch buffers become saturated in case of bursty traffic conditions where none of the policies have any influence in managing buffers more effectively.



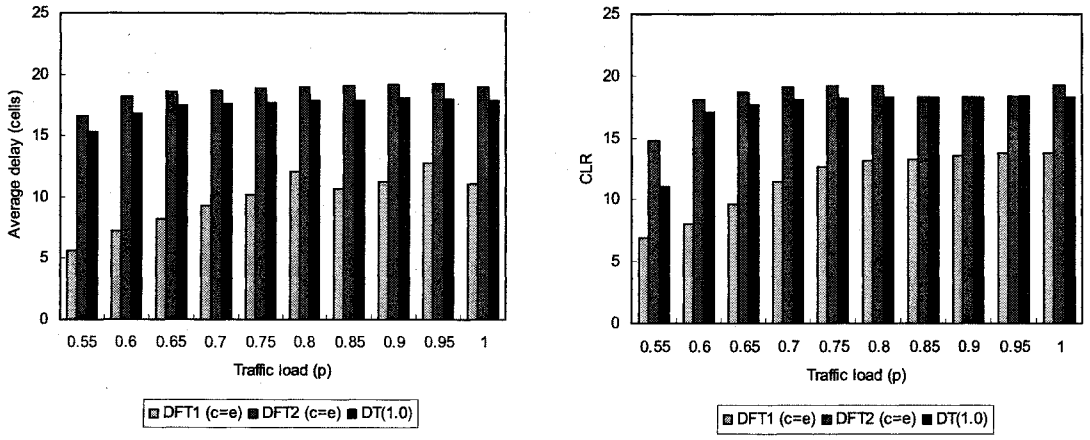
(a) (b)

Fig. 7.11 – Cell loss rate versus the traffic load for DFT1, DFT2 and DT policies: (a) h = 8 and (b) h = 12

In Fig. 7.11, CLR performances under varying traffic loads are considered for number of heavily loaded ports, h = 8 and 12. In both of these cases, the load on heavily

loaded ports is set as $L_p = 0.95$ and traffic load is given as $p = 0.8$. At moderate input load levels, DFT1 underutilises the buffer space resulting in high loss rates when compared with DFT2 and DT policies. Similar characteristics, but higher CLR, is observed when the number of hotspot output ports is increased to 12, see Fig. 7.11(b). For heavy traffic conditions all schemes achieve similar loss performances. Regardless of the traffic condition, DFT2 policy matches the cell loss performance of DT policy.

The average delay introduced by each policy for the same traffic conditions is shown in Fig. 7.12. In case of DFT2 policy, cells encounter more delay as they travel through the shared buffer switch architecture. As in the previous conduct, DFT1 policy is strict on queue lengths and likewise average delay inclines with the traffic load. On the contrary, DFT2 and DT policies provide fairly stable delays as the traffic load increases.



(a) (b)

Fig. 7.12 –Average delay versus the traffic load for DFT1, DFT2 and DT policies: (a) $h = 8$ and (b) $h = 12$

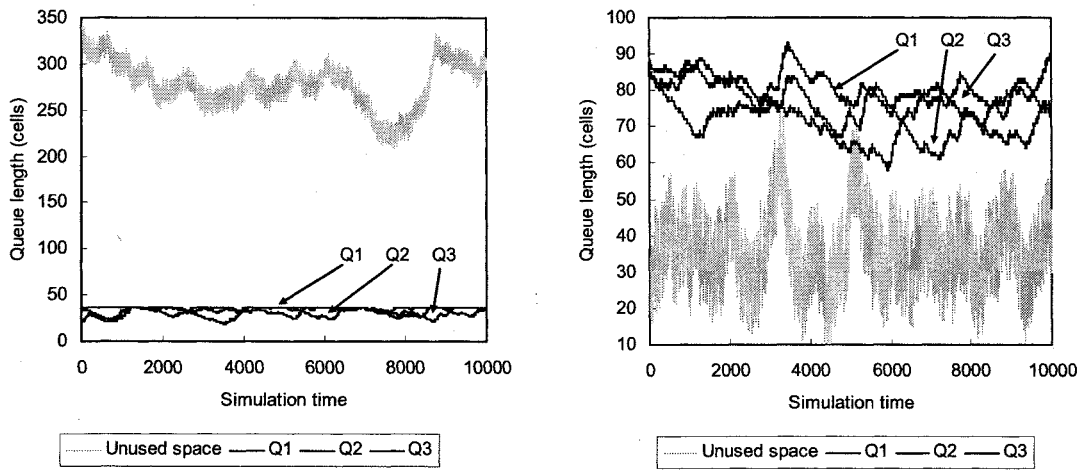


Fig. 7.13 – Queue lengths versus the simulation time for three heavily loaded ports, Q1, Q2 and Q3, and unused buffer space: (a) DFT1, and (b) DFT2

In Fig. 7.13, buffer occupancies of three heavily loaded ports, Q1, Q2 and Q3, and empty buffer space over 10,000 sampling points for both DFT1 and DFT2 policies are illustrated. Note that sampling rate is the same as transient plots in Fig. 7.8. In this configuration, traffic load, the number of heavily loaded ports, and the load on each heavily loaded port are given as $p = 0.8$, $h = 3$ and $L_p = 0.95$. Transient behaviour of DFT1 policy has a resemblance to PSPP where a large portion of buffer space is set aside for lightly and moderately loaded output ports considering that at some stage they will become active and start claiming buffer space from heavily loaded ports. Due to extent of spared buffer space, heavily loaded ports cannot make use of these buffers. In this setup, queue lengths of active queues are almost as low as fair share size, e.g. complete partitioning. Noting this fact, one can easily justify that the loss performance of DFT1 policy can be characterised as nearly output queueing depending on the amount of buffers accommodated for heavily loaded ports. On the other hand, DFT2 policy is reactive and considerably more dynamic. It treats each

queue (both active and inactive) individually; hence it is possible to observe frequent fluctuations in queue lengths and unused buffer space. Treating each queue on individual basis, on each cell arrival, offers finer thresholds and makes the shared buffer pool more visible across output ports. All heavily loaded ports receive a fair share of the buffer space leaving enough space for lightly and moderately loaded ports to grow freely during transient states. In contrast to DT policy, more buffer space is allocated by DFT2 to heavily loaded ports, whereas DT policy allocates space equivalent to unoccupied buffers when $\alpha = 1.0$.

7.3. Multicast Traffic

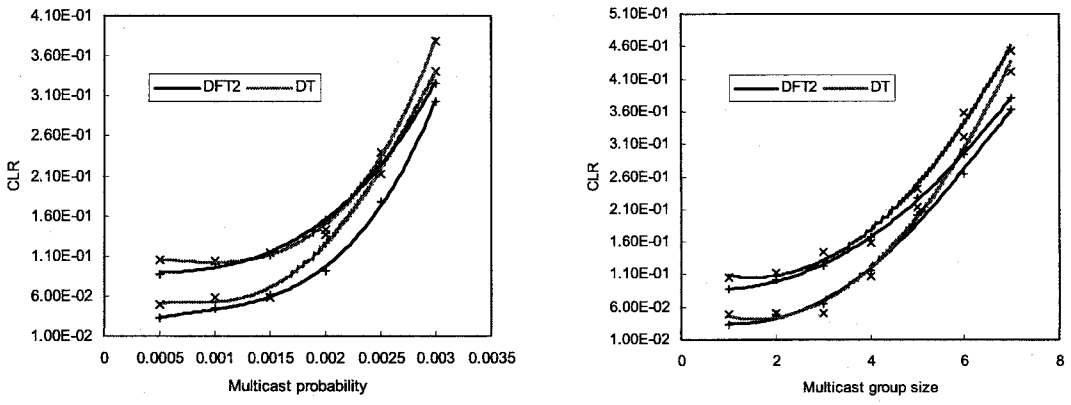
In this section, performances of DFT2 policy is compared with the DT policy under multicast traffic conditions. In multicast traffic, on arrival replicates of a cell are produced and sent to the corresponding output queues. At this stage, the threshold policy decides whether to accept or reject incoming replicates.

Table 7.6 – Parameters for multicast simulation

Parameter	Value
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Number of heavily loaded ports (h)	{1, 3}
Load on heavily loaded ports (L_p)	0.95

Simulation parameters are given in Table 7.6. Two situations considered are the number of heavily loaded ports given as $h = 1$ and 3. Figure 7.14(a) depicts the cell loss rate performances of DT and DFT2 policies under multicast traffic for various multicast probabilities with multicast group size set to one showing exponential

characteristics. Multicast probability is the rate at which replicates are produced during each arrival and multicast group size is the number of cells delivered to different output ports after each multicasting process. As the multicast probability increases, switch suffers from high cell loss rates due to high volume of traffic injected into the system. DFT2 policy demonstrates better cell loss performances than DT policy. For moderate multicast probability when h = 3, there is an overlap between DT and DFT2 due to exhaustive traffic introduced by multicasting and the number of heavily loaded ports. Both policies cannot cope with this intensity of traffic; hence there is a marginal difference among two policies.



(a) (b)

Fig. 7.14 – Cell loss rate performance for DT and DFT2: (a) multicast probability, and (b) multicast group size, for h = 1 and h = 3

A similar situation is shown in Fig. 7.14(b). In this case multicast probability is fixed at 0.0005 and group size is varying from one to seven. As the group size increases, switch outputs receive cells with higher numbers. This results in higher cell loss rates, as switch buffers cannot accommodate high volume of traffic. DFT2 policy offers a finer control by employing dedicated thresholds and tries to accommodate replicates

temporarily albeit the limited buffer space. Whereas DT policy employs a global threshold where all queues are treated alike. In this way, relatively short queues cannot benefit from unused buffer space immediately and their arrivals are rejected. Note that DT policy can be optimised according to the requirements of these circumstances by adjusting the share parameter appropriately.

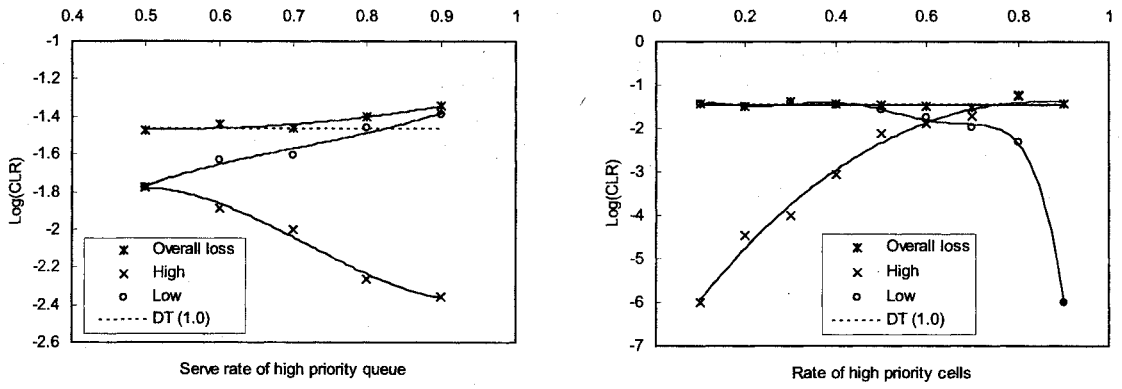
7.4. Quality of Service Traffic

The aim of this simulation study is to demonstrate how DFT policy (i.e. DFT2) copes with priority traffic by using per-queue thresholds. Per-queue thresholds are applied to priority queues individually to ensure that active queues do not dominate the usage of shared buffers, hence each priority queue has access to buffers at any time. During simulations, two priority classes have been considered, namely high and low. High and low priority cells are sorted into corresponding queues on arrival to output port buffers and served by using weighted round robin (WRR) scheduling algorithm. High priority queues are provided with higher proportion of bandwidth than low priority. The scheduling algorithm ensures that none of the queues are starved. Simulation parameters are given Table 7.7. Note that buffers are completely shared among $2N$ priority queues.

Table 7.7 – Parameters for QoS simulation

Parameter	Value
Network size (N)	32
Buffer size (M)	640 cells
Traffic load (p)	0.8
Mean burst length (T_{ON})	100 frames
Arrival rate (λ)	100%
Number of heavily loaded ports (h)	1
Load on heavily loaded ports (L_p)	0.95

In Fig. 7.15, the implications of different service and arrival rates of high priority cells are illustrated. Results are shown in logarithmic scale and overall loss is obtained by summing CLR of high-priority and low-priority cells. Overall CLR of DFT2 policy is compared with dynamic thresholds policy to observe the impact of employing per-queue thresholds on best-effort performance. In Fig. 7.15(a) both high and low priority queues receive traffic at the same proportion, i.e. 0.5. As one might expect when high priority queue serve rate is at 0.5, CLR of both priority classes are the same. As the serve rate of the high priority queue increases, more cells are served from the head of line of the high priority queue. Fewer cells are removed from low priority queues and therefore queue lengths increase dramatically. In such circumstances, the buffer management policy is in effect and rejects excess low priority cells. Meanwhile overall CLR is nearly stable when compared with DT policy. When the proportion of bandwidth allocated for high priority cells becomes very high, stability of overall CLR is disrupted.



(a) (b)

Fig. 7.15 – Cell loss rate performance of priority traffic: (a) serve rate of high priority queues, and (b) arrival rate of high priority cells

The effect of arrival of high priority cells is depicted in Fig. 7.15(b). In this setup, bandwidth allocated for high priority queues is 75%. CLR of high priority cells increases together with arrival rate of this type of traffic. Hence, buffer management unit regulates the queue lengths to prevent domination of high priority queues and avert idling of low priority queues. In Figs. 7.15(a) and 7.15(b), the overall CLR performance is compared with the DT policy. Overall performance of the integrated buffer management and scheduling policy is approximately the same as that of the DT policy. Nonetheless, low priority queues must be served at sufficient rates to maintain stable overall loss performance. Due to ease of implementation of per-queue thresholds and decoupling buffer management and bandwidth scheduling schemes, any scheduling algorithm can be adopted in the proposed framework. While DFT policy allocates buffer memory to priority queues, the scheduling algorithm allocates bandwidth independently. The integrated buffer management and scheduling framework tries to ensure that best-effort traffic performance is not affected by the way priority queues are served.

7.5. Summary of Results

In this chapter, comparative study of dynamic buffer management policies, DT, DQT, PSPP, DFT1 and DFT2, was carried out via simulations. A summary of main features of these policies can be found in Table 7.8. Firstly, control parameters of existing threshold schemes, namely DT, DQT and PSPP, were optimised under given network traffic conditions. Provided results are restricted to traffic model used in this study. Nevertheless, more detailed investigation of these control parameters are studied in the corresponding references (see studies by Choudhury and Hahne (1998), Fan et al.

(1999) and Chu et al. (2002)). Secondly, control parameter for DFT policies were tuned in a similar way. Obtained results show that control parameter must be set as c = e to offer optimal cell loss performance. Note that, finer tuning might be needed depending on the loss and delay requirements as well as traffic patterns considered. Thirdly, cell loss and average delay performances of DT, DQT and PSPP were contrasted on the basis of best-effort traffic. Results show that DT policy achieves low loss, as it has both reactive and dynamic features. That is, short term changes in traffic conditions are captured instantly, and empty buffer space fluctuates continuously capturing the long term changes in traffic phase. DQT achieves similar cell loss performances as DT policy by offering slightly lower average delay. PSPP is a restrictive policy where a large portion of buffer space is left aside limiting the lengths heavily loaded ports. By this way, delay offered by this policy is smaller with the expense of higher cell loss rate.

Following a similar approach, performances of DFT1 and DFT2 were analysed in contrast to DT policy. Like PSPP policy, DFT1 is restrictive and it is not very reactive to changes in traffic phase resulting in poor utilisation of buffers. On other hand, DFT2 matches the cell loss rate performance of DT. This policy is less restrictive on queue lengths sparing sufficient buffer space for lightly and moderately loaded ports to grow freely. DFT2 policy outperforms DT under multicast traffic by achieving low loss. Finally, it was shown that DFT2 policy supports priority traffic by employing per-queue thresholds, where each priority queue is treated individually. Proposed integration framework for buffer management and scheduling is made up of loosely coupled dynamic buffer management and scheduling algorithm blocks. Obtained

results show that DFT2 policy ensures that best-effort cell loss performance is maintained regardless of the way bandwidth is allocated to priority queues.

Table 7.8 – Summary of dynamic threshold policies

Policies	Optimal control parameter value	CLR performance	Delay	Reactive	Support for priority traffic
DT	$2^{-1} - 2^{1}$	Good	Moderate-High	Yes	Possible
DQT	1.0	Good	Moderate	No – Persistent threshold	Multi-level thresholds
PSPP	1.5 – 2.0	Moderate	Low	Yes	No
DFT1	e	Poor	Low	No	No
DFT2	e	Good – Improved multicast performance	High	Yes – Dedicated threshold	Per-queue thresholds
		1 Good 2 Moderate 3 Poor	1 High 2 Moderate 3 Low		

CHAPTER 8

CONCLUSIONS AND FURTHER WORK

8.1. Conclusions

In this chapter, a summary of discussions of this research study is presented, contributions of the study are highlighted and finally a discussion on potential future work is provided.

In this research study, buffering in high-performance packet switch architectures has been considered. Buffering in packet switches can be categorised as electronic and optical. Switch architectures with electronic memories are classified according to the positions of their FIFO buffers. These are mainly input queuing, output queuing, virtual output queueing (VOQ) and shared buffering. Among all these approaches output queuing and shared buffering achieve optimal throughput-delay performance under uniform input traffic. Switch architectures with optical buffers are categorised according to the positions of optical delay line (ODL) buffers, namely single-stage feedforward, single-stage feedback, multistage feedforward and multistage feedback. Performance considerations of ODL based switches are not only limited with network performance. For instance, optical signal power is subject to degradation due to continuous circulations within the ODLs.

Scalable packet switch architectures with distributed buffers can be achieved by employing buffered nodes in multistage interconnection network (MIN) architecture. In Chapter three, the performance implications of different buffering approaches

(including 2×2 ODL switch architectures) were presented and evaluated by means of simulations. It was shown that the output buffering and the shared buffering achieved improved results in terms of throughput-delay ratio and packet loss rate under general traffic conditions. It was shown that the cascaded optical delay line switch architectures achieve similar packet loss and throughput performances as the shared buffering. Nevertheless, the ODL switches can only be implemented in a slotted system where packets have fixed-lengths and either one or no packets arrive in a time-slot. Besides, the system performance, such as signal power, is another concern as packets travel from inputs to outputs of the all-optical MIN; hence making large switch size implementation a challenging task.

All buffering schemes, including shared buffer switch, suffer from high packet losses due to imbalanced input traffic. In Chapter four, the impact of imbalanced traffic on shared buffer switch architecture was analysed for two cases, complete sharing and complete partitioning. Buffer management in shared buffer switches is mainly categorised as static threshold, adaptive control, push-out (preemptive) and dynamic threshold with the latter policy offering relatively good performance in terms of cell loss rate and average delay with little control overhead, whereas preemptive policies offer the best performance at the cost of complex implementation. In this study an improved dynamic threshold policy, decay function threshold (DFT) policy, was proposed (see Chapter five), implemented (see Chapter six) and evaluated (see Chapter seven) by means of numerical simulations. Results obtained show that DFT policy is as good as the dynamic threshold policy proposed in (Choudhury and Hahne 1997) in terms of cell loss rate, however, it offered better cell loss performance than dynamic thresholds policy when multicast traffic was considered (see Chapter seven).

An integration framework that couples buffer management and bandwidth scheduling was also proposed in this study. This framework employs per-queue thresholds to regulate priority queue lengths whilst the scheduler attempts to allocate bandwidth by employing a scheduling algorithm such as weighted round robin (WRR), weighted fair queueing (WFQ) etc. Simulation results achieved show that the overall cell loss rate of quality of service (QoS) traffic is similar to the best-effort traffic performance of dynamic threshold policy proposed in (Choudhury and Hahne 1997).

8.2. Further Work

As an outcome of this study, two possible paths have been identified in this field of study for further research: (i) Intelligent switch control and (ii) all-optical packet switches with randomly accessible optical memory. First path aims to utilise artificial intelligence techniques in control of packet switches and the latter benefits from the recent developments in controlling the speed of light pulses.

8.2.1. Intelligent networks: Adaptive learning control

Adaptivity and intelligence of control can be improved by employing soft-computing techniques such as artificial neural networks (ANN), fuzzy logic (FL), and machine learning techniques. ANN is inspired from biological nervous system where highly interconnected processing elements work together to solve a problem. Fuzzy logic is an extended form of Boolean logic that can cope with partial truth. Machine learning is concerned with the ability of learning from previous experience. Now let's review some of the applications of these techniques in buffer management and packet scheduling. ANN approach proposed in (Yousefi'zadeh 2002) aims to reduce the

packet loss in multiple source queueing system by means of prediction feature of neural networks. Fixed structure feed-forward neural network is trained in the presence of self-similar traffic to cope with packet loss and fairness issues. An alternative approach, fuzzy buffer management scheme, was proposed in (Ascia et al. 2002). In this approach, thresholds are adapted for each queue according to the traffic conditions by using fuzzy inferences. Membership functions are then selected and optimised via Genetic Algorithms (GAs). A machine learning approach applied to packet scheduling in routers was suggested in (Ferra et al. 2003). In this study, reinforcement learning (RL) scheme is used to effectively learn scheduling policies to satisfy QoS requirements. The main benefit of this approach is that RL scheme can adapt varying traffic conditions and QoS requirements without the help of any external supervisor. RL is widely used in packet routing decision (see studies by Boyan and Littman (1994) and Dorigo et al. (1996)). A novel concept that utilises intelligent techniques was reported in (Gelenbe et al. 2004), known as "cognitive packet network" (CPN). This approach suggests that routing and flow control intelligence are moved from network node to "smart" packets, also known as cognitive packets.

Based on various studies that utilise intelligent methods, it is essential to use such techniques to improve the capability of existing buffer management policies and remove the human factor, e.g. share parameter tuning etc. For this reason, it is worthwhile to initialise a framework, which essentially benefits from adaptive learning, to address the buffer management problem. Here, the adopted buffer management approach employs a type of machine learning technique, known as stochastic learning automata (SLA) (Sutton and Barto 1998), which iteratively gathers

information about its environment and adopts a threshold policy over time. Essentially this approach is an agent-based system. After a set of interactions with its environment, a reward is returned back to the agent. This reward is then compared with a reference reward which is normally weighted average of previous rewards. This reward scheme helps to determine favourable packet admission probabilities and offers improved performance in time. A detailed description of the proposed agent-based model and preliminary results are given in Appendix C. The proposed intelligent buffer management approach is open to further developments. For instance, it is worthwhile modelling a robust reward scheme so that the intelligent agent is more capable of efficiently realising the changes in traffic conditions. Moreover, a weighted sum of metrics such as throughput and average delay can also be used. Instead of a single agent, multiple agents (one for each queue) can be employed where each agent receives its own reward back from the environment rather than a global one.

8.2.2. All-optical packet switches with optical random access memory

Recent studies have shown that it is possible to control the speed of light by using off-the-shelf components. In (Gonzalez-Herraez et al. 2005), the authors demonstrated a method to control optical signals as they traverse along an optical fibre. By this way, high-speed optical signals can be slowed down for header processing; hence this does not require costly conversion from optical to electrical and vice-versa. Besides, packets can be optically stored to ease network congestion problems. Lately, IBM researchers were able to actively control light pulses on a chip by using photonic crystal waveguides and applying voltage across the waveguide (Vlasov et al. 2005). When heated, the waveguide changes the refractive index; in turn this tunes the speed

of the light pulses. Higher refractive index slows down those light pulses that travel in the waveguide.

Considering recent advancements in photonics, optical switch architectures discussed in Chapter two of this thesis must be re-visited. Design patterns used in electronic switches can be readily applied in development of all-optical switch architectures with optical buffers. Lengthy delay lines are replaced with on-chip optical buffers which makes all-optical switch-on-chip (SOC) architectures viable. Both physical and operational performances must be considered in such design practices. For instance, optical buffers can be completely shared among output ports and dynamic buffer management approaches discussed in this thesis can be utilised to boost the switch performance. Nevertheless, the development of all-optical switch architectures with optical buffers requires further research in some other areas, such as (i) all-optical header processing, (ii) all-optical control logic, and (iii) randomly accessible optical memory.

APPENDIX A – Queueing Theory

Little's law (Gross and Harris 1998): Let $E(L)$ be the mean number of customers in the system, E(S) be the mean sojourn time and λ be the number of customers entering the system per unit time. Hence, Little's formula is given as $E(L) = E(S).\lambda$.

Pollaczek-Khrincin formula (Gross and Harris 1998): Mean waiting time, W, is the time customer has to wait for service. Therefore expected waiting time is given as $E(W) = E(N).E(S) + E(R)$, where $E(N)$ is the expected number of customer waiting for service, $E(S)$ is the expected mean service time, and $E(R)$ is the expected remaining work currently in the server, where R is the residual service time.

Single-server queue models (Gross and Harris 1998):

λ: arrival rate μ: service rate p: utilisation ($p = \lambda/\mu$) σ: standard deviation $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$	L: average number of customers in the system L_q: average number of customers in the queue w: average time spent in the system w_q: average time spent in the queue

M/G/1	M/M/1	M/D/1
$L = p + \frac{p^2(1+\sigma^2\mu^2)}{2(1-p)}$	$L = \frac{p}{1-p}$	$L = p + \frac{1}{2}\frac{p^2}{1-p}$
$L_q = L - p$	$L_q = \frac{p^2}{1-p}$	$L_q = \frac{1}{2}\frac{p^2}{1-p}$
$w = L/\lambda$	$w = \frac{1}{\mu(1-p)}$	$w = \frac{1}{\mu} + \frac{1}{2}\frac{p}{\mu(1-p)}$
$w_q = w - \frac{1}{\mu}$	$w_q = \frac{p}{\mu(1-p)}$	$w_q = \frac{1}{2}\frac{p}{\mu(1-p)}$

APPENDIX B – Benchmarking

In order to validate the developed simulation program in this study, a simple comparison study is carried out. Basically, this involves comparing simulated results in this study with the results in the literature. In (Choudhury and Hahne 1997), a comprehensive study was carried out by Choudhury and Hahne to demonstrate the robustness of their dynamic threshold policy. These results are more established when compared with other studies (Ascia et al. 2002; Chu et al. 2002; Fan et al. 1999; Kesselman and Mansour 2002) and therefore these can be used as benchmark results noting the assumptions made in (Choudhury and Hahne 1997) as well as in this study.

Figure B.1 illustrates cell loss rate (CLR) versus traffic load (p) results for static threshold (ST) and dynamic thresholds (DT) acquired from (Choudhury and Hahne 1997), i.e. benchmark results. This figure also includes simulated results for DT policy produced in this study. Note that CLR dimension is in logarithmic scale. Benchmark results are based on the following assumptions. ST scheme assume a threshold value T equal to 120 cells and DT scheme assumes share parameter α equal to one. In (Choudhury and Hahne 1997), it is also assumed that traffic source model is On-Off with on and off periods geometrically distributed. Switch dimensions and buffer size are 512×512 input/output ports and 6500 cells respectively, providing a fair share size of 12 cells per output port. Whereas in this study traffic source is modelled as IPP and a fair share size of 20 cells per output port is considered. As can be seen from the figure, simulated results are quite similar to benchmark results due to similar assumptions made in this study. The difference between benchmark DT and

simulated DT is that a fair share size of 20 cells per output simulated DT provides more buffer space for individual bursts resulting in smaller CLR.

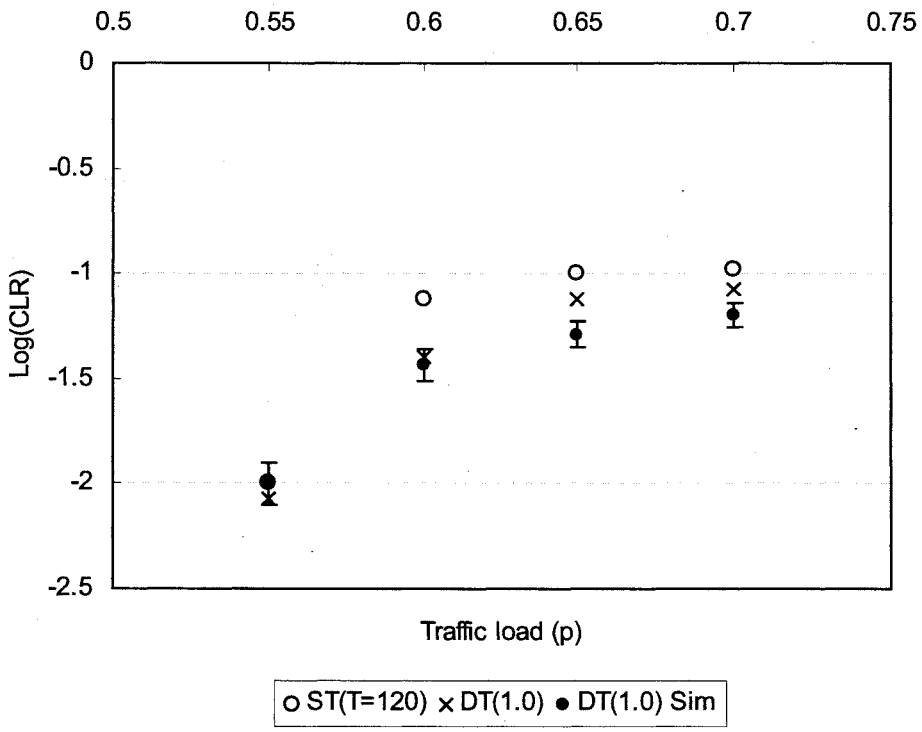


Fig. B.1 – Cell loss rate versus traffic load for static thresholds ($T = 120$) and dynamic thresholds ($\alpha = 1$)

APPENDIX C – Agent-Based Buffer Management

There exist two possible actions associated with a FIFO queue, which are admit and block. Each action has an action probability and the FIFO queue switches between them continuously. In an $N \times N$ shared buffer switch, a received packet is admitted to queue i at time t with a probability $p_t^i(a)$ or rejected with a probability $p_t^i(b)$. These can be formally presented using the transition matrix below, where '0' denotes admit and '1' denotes block.

$$\begin{array}{ccc} & 0 & 1 \\ 0 & p(a) & 1-p(a) \\ 1 & 1-p(b) & p(b) \end{array} \qquad \text{(C.1)}$$

After every trial period, these action probabilities are updated to ensure the likelihood of selecting the most favourable action. If the selected action is good, then its probability is incremented by a faction (i.e. learning speed) β, given as:

$$p_{t+1}^i(a) = p_t^i(a) + \beta.[1 - p_t^i(a)] \qquad \text{(C.2)}$$

Whereas the probability of the other action is decremented as given by:

$$p_{t+1}^i(b) = p_t^i(b) + \beta.[0 - p_t^i(b)] \qquad \text{(C.3)}$$

At the end of a trial (e.g. batch of actions), a cumulative reward is returned back to the agent. In general terms, an agent uses its sensors to perceive its environment and acts through its effectors (Sutton and Barto 1998). The agent compares the new reward with a reference reward so that to determine whether previous actions were favourable or not. The cumulative reference reward can be given as:

$$\bar{r}_{t+1} = \bar{r}_t + \alpha.[r_t - \bar{r}_t] \,, \tag{C.4}$$

Where α denotes the step size and r_t is the current reward. In this study, reward can be denoted as the number of packets dropped between two successive sampling points $\Delta t = t_2 - t_1$. Therefore the total reward during Δt can be given as weighted sum of rewards for each queue:

$$r_{\Delta t} = \sum_{i=1}^{N} w_i r_{\Delta t}^i \tag{C.5}$$

An activation function is used to determine whether it is feasible to update the admission probabilities or stay with the current configuration. The aim of the learning automata is to find a set of packet admission probabilities for the shared buffer switch ultimately to reduce the overall packet loss. Proposed agent-based model is depicted in Fig. C.1.

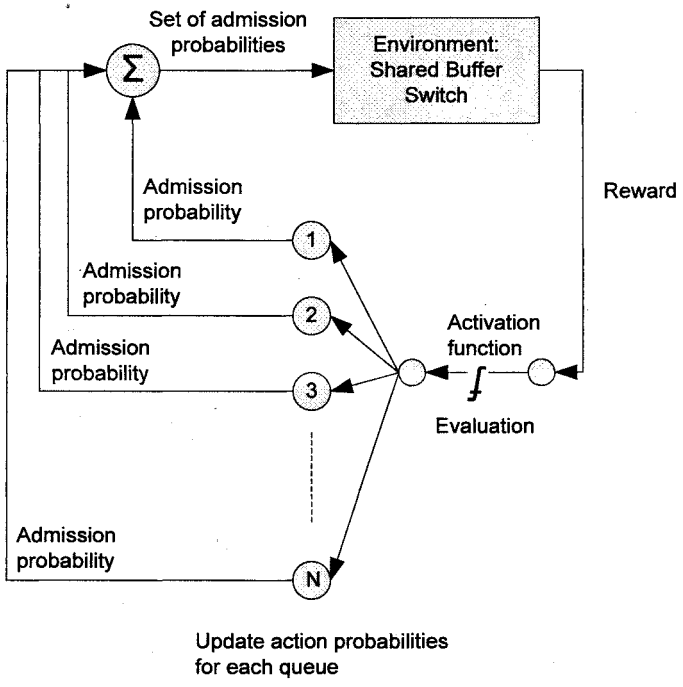


Fig. C.1 – Adaptive learning system model

A similar simulation model was used as described in Chapter six. During simulations, a 32×32 non-blocking switch with 640-cell storage capacity memory shared among 32 output ports is considered. In this setup, traffic load and various learning speeds are given as 0.8 and β = {0.01, 0.08, 0.15, 0.3} respectively. Cell loss rate (CLR) versus different learning speeds for the proposed scheme is illustrated in Fig. C.2. It is possible to observe that the adaptive learning policy with learning speeds $\beta = 0.01$ and $\beta = 0.3$ converges to a non-optimal loss performance. This is because, at $\beta = 0.01$ (i.e. a very slow learning speed) it takes longer for the adaptive learning policy to reach the desired loss performance. Whereas at $\beta = 0.3$, a relatively large step-size, the learning policy converges to a local optimal rather than a global one. For both $\beta = 0.08$ and $\beta = 0.15$, the learning policy successfully converges to an optimal loss performance. However, the convergence takes longer for $\beta = 0.08$. At the start of the simulation, CLR performance fluctuates temporarily because of traffic phase not being stabilised and the learning agent not having previous knowledge of the environment.

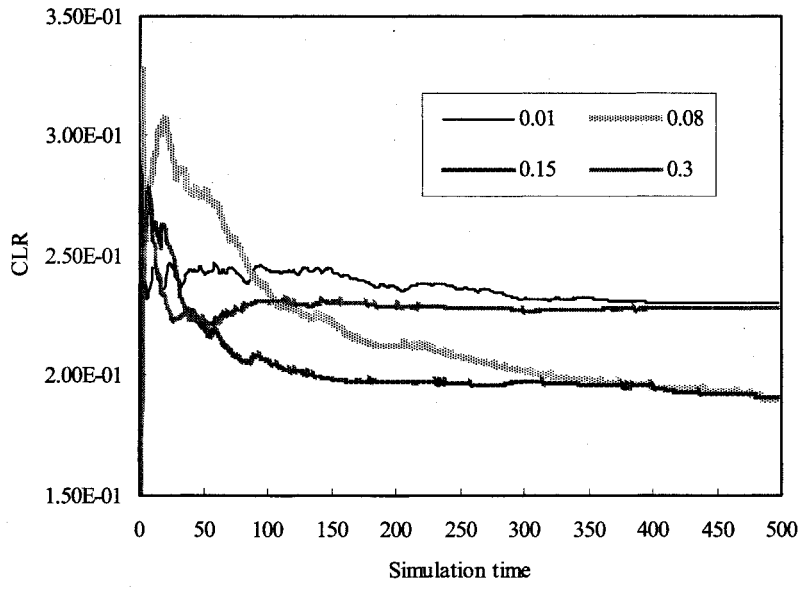


Fig. C.2 – Convergence: CLR versus simulation time for different learning speeds

APPENDIX D – Published Key Articles

Page 199

B. Gazi and Z. Ghassemlooy, "Exponential decay queue thresholds for optimal buffer sharing," Proceedings of The IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2005), Twenty-Third IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria, 15-17 February 2005, pp. 590-594.

Page 207

B. Gazi and Z. Ghassemlooy, "Performance characteristics of cascaded delay-line node architectures in single-path interconnection networks," International Journal of Communication Systems, vol. 17, no. 5, pp. 447-459, 2004.

EXPONENTIAL DECAY QUEUE THRESHOLDS FOR OPTIMAL BUFFER SHARING

Boran Gazi, Zabih Ghassemlooy
Optical Communications Research Group, The University of Northumbria, Newcastle upon Tyne, NE1 8ST, U.K. borangazi@ieee.org, fary.ghassemlooy@unn.ac.uk

ABSTRACT

In an N x N shared buffer switch architecture, a common buffer space is shared among output ports, which provides flexibility in resource allocation. However the problem arises when the buffer space is dominated by one or more output ports in case of imbalanced input traffic. Thus this results in an increased packet loss rates due to unfair sharing of buffer space. For this reason buffer management policies are essential to regulate the buffer space occupied by active output ports. In this paper we introduce a dynamic threshold policy, which is inspired from the exponential decay. This policy suggests that the buffer space accessed by an output port decays exponentially with respect to its queue length. In this work we compare the packet loss rate performance of this scheme with the dynamic thresholds policy by carrying out a simulation study. We have shown that the performance of the exponential decay function policy can be improved by considering unused buffer space. It is found that the improved policy achieves the same packet loss rate as the dynamic thresholds policy with an improved buffer utilisation. The results suggest that the best threshold based buffer management policy is the one that allows buffer overflow in the case of hotspot packets.

KEY WORDS

Shared Buffer Switch, Dynamic Buffer Management.

1. Introduction

In packet switching network nodes, packet losses mainly occur due to simultaneous arrival of packets at the same output port. In order to avoid losses, packets are stored and scheduled for later departure.

Buffering in packet switching network nodes is a common technique to overcome packet losses in case of bursty network traffic. A significant amount of work has been reported in this field of study to find an effective solution for the contention problem [1][2]. Common buffering strategies are input-queuing, output-queuing, recirculation buffering, and shared buffering. In [1] it has been shown that the shared buffering scheme is one of the best performers among other techniques as it provides a notion of flexibility when allocating buffer space for the incoming packets.

A shared buffer switch (SBS) consists of a common buffer space entirely shared among N output ports in an N x N non-blocking switch [2]. Despite its flexibility, SBS greatly suffers from inefficient use of shared buffer space when the input traffic is asymmetric (e.g. hotspot). The main reason for this is that heavily loaded output ports dominate the usage of shared buffer space and therefore lightly or moderately loaded output ports are unable to access the buffer space when required. For this reason a buffer management policy has to be adapted for efficient and effective use of buffer space in SBSs.

Buffer management policies can be categorised into two classes [2]: static and dynamic. In the former, the policy is based on static parameters set by using statistical information and repeated simulations, whereas in the latter the policy for controlling the common buffer space is based on the information from dynamic environment parameters. Dynamic policies outperform static policies, since they have higher awareness of network dynamics (e.g. traffic load and pattern, link failure, priority traffic etc).

Buffer management policies can be further categorised as preemptive and non-preemptive. In preemptive case, a packet already in the buffer is simply replaced [3]. Preemptive policies such as drop

on demand (DoD) [3] and push-out with thresholds (POT) [4] admit packets until when the loss is unavoidable. In both cases, the activities of active queues are controlled by dropping packets from the longest queues when a packet arrives to an inactive queue and finds a full buffer. These policies are naturally adaptive and utilise buffer space better than non-preemptive policies. However the main problem with these policies is that pushing out a packet already in the buffer is not feasible, and searching for the longest queue has the complexity of O(N) in an N x N SBS.

Non-preemptive policies use dynamic thresholds when admitting packets into the buffer. Dynamic threshold (DT) policy, proposed by Choudhury and Hahne, was developed to achieve the dynamism of DoD together with simplicity of the static threshold [5]. Dynamic threshold policy studied in [6] aims to ensure fair sharing of buffer space whilst enabling active queues to use any unused buffer space. Another dynamic buffer management policy, known as harmonic buffer management policy, uses a cumulative threshold for the total length of the k largest queues [7]. Partial sharing with partial partitioning (PSPP) policy divides buffer space into two virtually separated parts according to the number of active and inactive output ports [8]. Finally, threshold-based selective drop (TSD) scheme is inspired from the max-min bandwidth-sharing scheme [9].

In this paper we propose a new buffer management policy that is based on the exponential decay, called decay function threshold (DFT), and contrast its performance to the well-known DT by carrying out a simulation study. Our findings show that DFT offers similar performance as DT with improved utilisation of the buffer space. The following section provides a brief explanation of both DFT and DT schemes. Section 3 describes the simulation model. Simulation results are discussed in Section 4. Finally conclusions and paths for further developments are presented in Section 5.

2. Dynamic Buffer Management Policy

2.1 Decay Function Threshold (DFT)

The main issues concerning resource sharing among N consumers are fairness and resource utilisation that needs to be addressed when developing resource management policies. The main problem with shared buffer switches is that when one or more output ports are loaded more than others, they start utilising more buffer space, as more packets are addressed to these output ports. At some stage heavily loaded output ports make full use of the buffer space and thus preventing lightly loaded ports accessing the buffer space when required. Hence this results in low throughputs and long delays.

DFT policy attempts to limit the lengths of active queues by using a decay function. In other words, the amount of buffer space that an output can access decreases exponentially with its queue length. This can be formally presented by:

$$T_i(t) = \frac{M}{e^{\left\lfloor \frac{N.q_i(t)}{M} + k \right\rfloor}} \tag{1}$$

Where $T_i(t)$ and $q_i(t)$ represent the threshold and queue length at port i at time t respectively; M is the buffer size, N is the switch size, and k represents the share parameter ($k \geq 0$). A packet arriving at port i at time t is blocked if $q_i(t) > T_i(t)$. DFT policy uses a separate threshold for each output queue. For $q_i(t) \geq 0$,

$$\log_e\left(\frac{M}{T_i(t)}\right) \geq k \tag{2}$$

$$\frac{M}{T_i(t)} \geq e^k \tag{3}$$

For a very large k, we have a very small threshold. Therefore, assigning a value for k from 0 to 1 is appropriate. In our simulations $k = 1$. Another decay function threshold policy that we propose in this paper uses both current queue size of port i and the available buffer space at time t, given by:

$$T_i(t) = \frac{M}{e^{\left\lfloor \frac{q_i(t)}{B+1} \right\rfloor}} \tag{4}$$

Where B denotes unused buffer space. Unlike (1), (4) checks the available buffer space. Hence, for a very low unused buffer space, it is more likely that a packet arriving to a long queue will be blocked (Fig. 1). Throughout this paper the policies in (1) and (4) are named as DFT1 and DFT2, respectively.

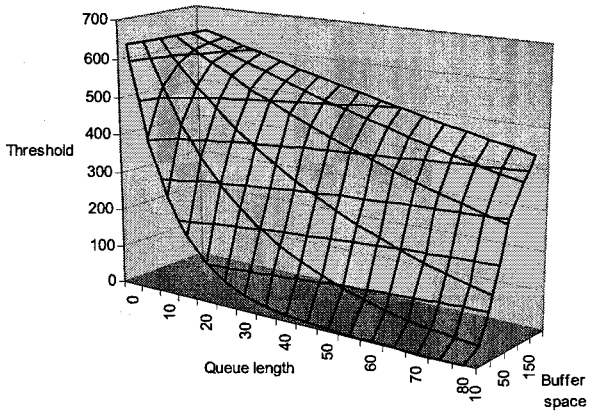


Figure 1 – DFT threshold against the queue length and buffer space

2.2 Dynamic Threshold (DT)

DT policy sets a single queue threshold for all queues that is proportional to the amount of unused buffer space. The aim of this scheme is to spare some buffer space for lightly loaded output ports queues to grow. Simply, a packet arriving at an output port is rejected if the queue length of that output port exceeds the threshold value at time t, $T(t)$:

$$T(t) = \alpha.(M - \sum_{i=1}^{N} q_i(t)) \tag{5}$$

Where α is the share parameter. The studies have shown that for different traffic conditions an α value between 2^{-1} and 2 provides better performance characteristics [2][5].

3. Simulation Model

In order to measure and analyse the performance characteristics of both DFT and DT policies, a simulation program is developed for parallel virtual machine (PVM) platform. The aim of using this platform is to achieve a realistic network environment with asynchronous and independent working processes (e.g. traffic generators, queue controllers, buffer management unit etc).

3.1 SBS Model

An N x N SBS consists of a buffer management unit that is responsible from storing buffer state, allocating space for newly arrived packets and calculating the threshold value on basis of a buffer management policy. Threshold value is updated at every space request from the queue controllers.

Queue controllers (there are N controllers in an N x N switch) accept newly arrived packets and service packets at the head of the queue. Based on the threshold value, packets are either granted buffer space or are rejected. Each queue controller serves one packet per frame-time[1] (i.e. deterministic service time). One-packet space is returned to the buffer management unit to update its state whenever a packet leaves the switch.

3.2 Traffic Model

Inputs of SBS are connected to independent asynchronous traffic generators each of which generates an input traffic according to a distribution. Packet lengths are fixed and at most one packet is generated during a frame-time. Each traffic generator is a realisation of interrupted Poisson process (IPP) with ON and OFF states [10]. The time spent in ON and OFF states is exponentially distributed with mean durations of T_{ON} and T_{OFF} respectively. Arrivals occur only at ON states with rate λ. Thus the traffic load can be formally presented by:

$$p = \frac{\lambda . T_{ON}}{T_{ON} + T_{OFF}} \tag{6}$$

Traffic distributions considered in our simulations is asymmetric. That is, each arrival packet has a probability r/N of being addressed to a particular output port (e.g. hotspot port). Hence, parameter r measures the degree of imbalance of the traffic distribution.

4. Results and Discussions

In this paper we study the performances of decay function threshold policies (DFT1 and DFT2) that are represented in Equations (1) and (4), and compare their performances with dynamic threshold scheme.

In our simulations we considered 32 x 32 non-blocking switch with 640-packet storage capacity shared among 32 output ports. Throughout this paper DT share parameter α is set to 1.0.

In Fig. 2, packet loss rate (PLR) performances of DFT1, DFT2 and DT policies are compared for 0.95 and 1.10 hotspot loads. Traffic load is 0.8 and T_{ON} = 100 frame-times. For both of the hotspot loads it can be seen that DFT1 performs worse than the other two when the number of hotspot ports is fewer. DFT1 policy does not consider the available buffer space. Even if there is enough buffer space for allocating newly arrived hotspot packets; these packets are dropped to spare space for inactive queues. In other words, DFT1 policy is unable to utilise the buffer space effectively when there are fewer number of active queues. As the number of hotspots increases, all three policies achieve approximately the same PLR performance.

In Figs. 3 and 4, we considered PLR performances for varying traffic loads. In both cases, hotspot load is set to 0.95 and T_{ON} is 100 frame-times. At moderate traffic load levels, DFT1 underutilises the buffer space which results in high loss rates when compared to both DFT2 and DT (Fig. 3). Similar characteristics but higher PLR is observed when the number of hotspot output ports is increased to 12, see Fig. 4.

DFT1 policy applies the threshold continuously regardless of whether there is available buffer space to allocate for the incoming traffic bursts. Thus, some buffer space is wasted at moderate traffic loads or slightly asymmetric traffic. In order to overcome this problem, DFT2 policy considers available buffer space. For example, consider a situation where there is only one packet space available, B = 1. Hence buffer is divided into e^c partitions, where c = $q_i(t)/2$. Therefore, if the queue length is short ($q_i(t) > 1$), this one-packet space can still be accessed.

[1] Frame-time is the duration of a packet

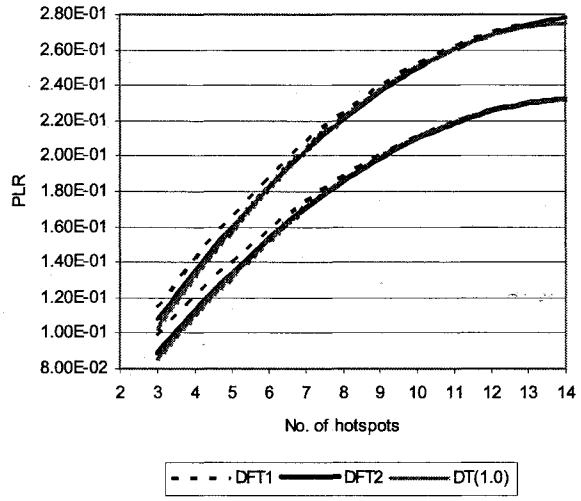


Figure 2 – Packet loss rate against the number of hotspots for DFT1, DFT2 and DT(1.0)

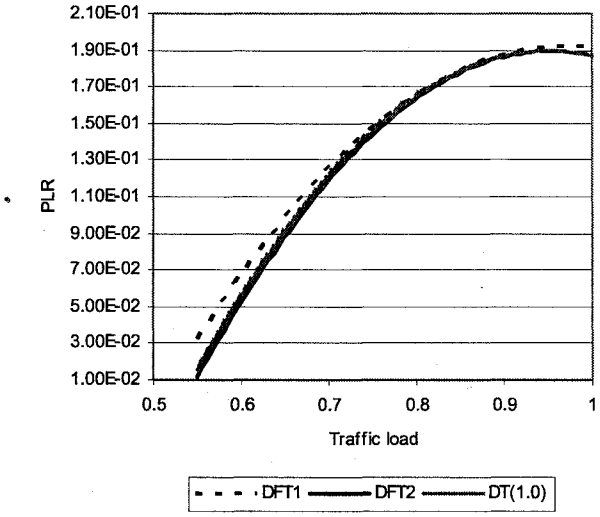


Figure 3 – Traffic load against the packet loss rate (8 hotspot ports with 0.95 hotspot load)

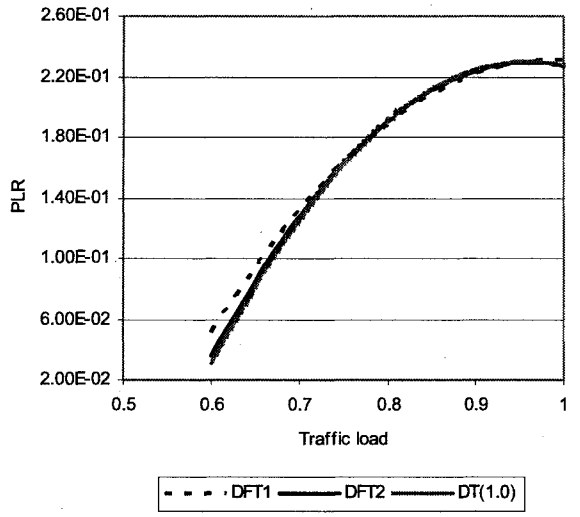


Figure 4 – Traffic load against the packet loss rate (12 hotspot ports with 0.95 hotspot load)

In Figs 5 and 6, we considered the buffer occupancies of hotspot ports and unused buffer space under asymmetric input traffic for a given duration of simulation. As can be seen from Fig. 5, the unused buffer space is rather stable when compared with Fig. 6. Moreover, the buffer space that DFT2 spares for inactive queues is less than DT. At some stage, available buffer space is as low as 10 packets. On the other hand, DT policy regulates the unused buffer space according to the activity of active queues. When $\alpha = 1$, DT policy spares a space which is almost the same as one hotspot port queue length.

Fig. 7 illustrates buffer utilisation against the traffic load for 8 hotspots with 0.95 hotspot load for all three policies. DFT1 and DFT2 display the worst and the best performance, respectively, for all traffic loads. DFT1 employs a very strict threshold scheme where hotspot port activities are limited based on the fair share level. DFT2 utilises the buffer best in all cases. Unlike DT policy, when the buffer space is very low, only very lightly loaded queues can still admit packets in DFT2 policy. However, DT policy spares guaranteed buffer space at all times. DFT2 policy allows buffer overflows and tries to accommodate tailored buffer space for lightly loaded output port packets. Therefore it is likely that non-hotspot packets may find the buffer full on arrival. This might be the reason for achieving almost the same PLR performances as DT but with an improved buffer utilisation. Therefore, based on these findings, the best dynamic threshold policy is the one that allows buffer overflow only in case of hotspot packets.

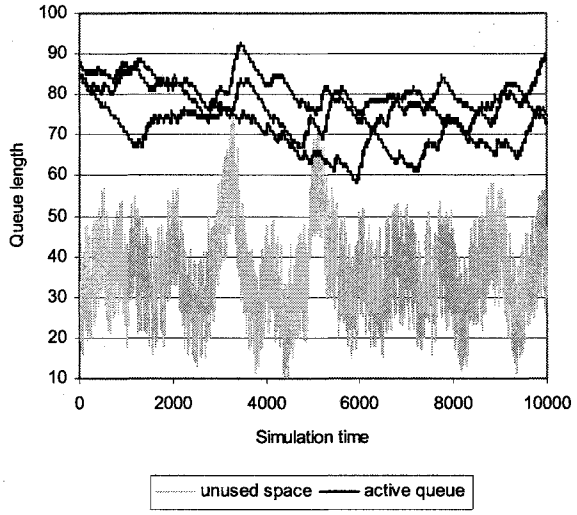


Figure 5 – Queue length against time for DFT2 (3 hotspots ports with 0.95 hotspot load)

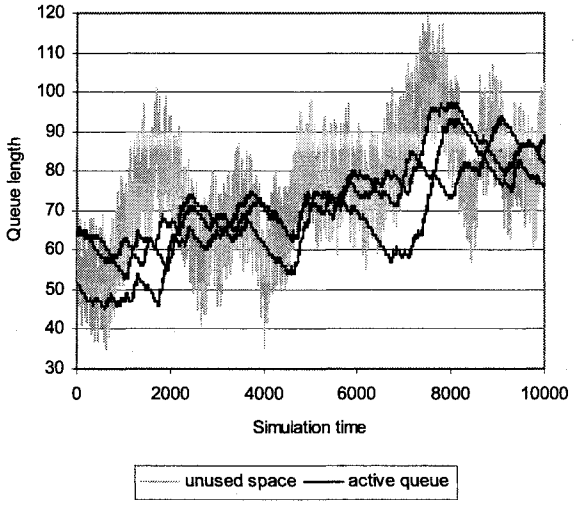


Figure 6 – Queue length against time for DT (3 hotspot ports with 0.95 hotspot load)

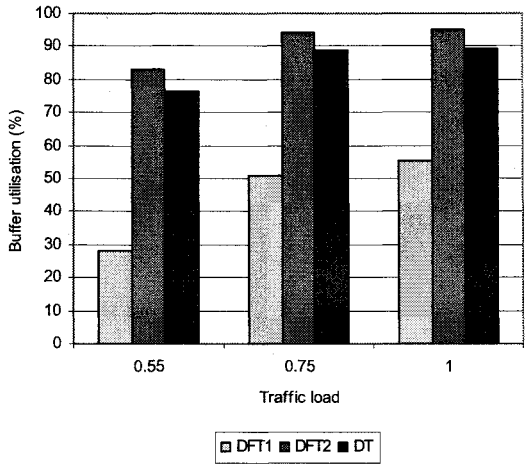


Figure 7 – Buffer utilisation against the traffic load (8 hotspot ports with 0.95 hotspot load)

5. Conclusion

In this paper we proposed two new buffer management policies, known as decay function threshold, for shared buffer switches and compared their performances with well-known dynamic thresholds policy. DFT policies suggest that the buffer space that an output port can access decreases exponentially with the queue length at that output port.

The results have shown that DFT1 policy underutilises the buffer space when the traffic load is moderate or slightly asymmetric. This is due to its strict and inflexible threshold policy. Threshold is applied to active queues continuously even when there is enough space to allocate hotspot streams. Hence, DFT2 policy was developed to resolve this problem. This policy states that if the current available buffer space is at its critical level, only very lightly loaded output ports with very short queue lengths can have access to it. When compared with DT policy, DFT2 achieves similar packet loss rates and offers slightly better buffer utilisation. The main reason for this is that DFT2 policy allows buffer overflow, whereas DT policy prevents overflow at all times. In other words, the buffer space that DFT2 spares for inactive queues is slightly less than DT policy. The main advantage of DFT policies is that, unlike DT, there isn't any static parameters set according to the network variables. Also our provisional findings suggest that an optimal dynamic threshold policy allows buffer overflows only in case of active queue packets (e.g. hotspot packets).

None of the dynamic buffer management policies consider scheduling and bandwidth allocation. Existing buffer management policies can only offer best-effort service at the switch outputs. In order to provide an improved quality of service [11] (e.g. priority traffic, fault tolerance in individual streams), and an efficient and effective buffer sharing, both buffer management and scheduling policies have to be considered in the same framework.

References:

[1] M. G. Hluchyj & M. J. Karol, Queueing in high-performance packet switching, *IEEE Journal on Selected Areas of Communications, 6*(9), 1988, 1587-1597.

[2] M. Arpaci & J. A. Copeland, Buffer management for shared-memory ATM switches, *IEEE Communications Surveys & Tutorials, 3*(1), 2000, 2-10.

[3] S. X. Wei, E. J. Coyle, & M.-T. T. Hsiao, An optimal buffer management policy for high-performance packet switching, *IEEE GLOBECOM*, Phoenix, AZ, 1991, 924-928

[4] S. Sharma & Y. Viniotis, Optimal buffer management policies for shared-buffer ATM switches, *IEEE/ACM Transactions on Networking, 7*(4), 1999, 575-587.

[5] A. K. Choudhury & E. L. Hahne, Dynamic queue length thresholds for shared-memory packet switches, *IEEE/ACM Transactions on Networking, 6*(2), 1998, 130-140.

[6] R. Fan, A. Ishii, B. Mark, G. Ramamurthy & Q. Ren, An optimal buffer management scheme with dynamic thresholds, *IEEE GLOBECOM*, Rio de Janeiro, Brasil, 1999, 631-637.

[7] A. Kesselman, & Y. Mansour, Harmonic buffer management policy for shared memory switches, *IEEE INFOCOM*, New York, 2002, 1615-1622.

[8] Y. S. Chu, J. P. Yang, C. S. Wu & M. C. Liang, Partial sharing and partial partitioning buffer management scheme for shared buffer packet switches, *IEICE Transaction on Communications, E85-B*(1), 2002, 79-88.

[9] J. P. Yang, Y. S. Chu & M. C. Liang, Threshold-based selective drop for shared buffer packet switches, *IEEE Communications Letters, 7*(4), 2003, 183-185.

[10] A. Adas, Traffic models in broadband networks, *IEEE Communications Magazine, 35*(7), 1997, 82-89.

[11] H. Fattah & C. Leung, An overview of scheduling algorithms in wireless multimedia networks, *IEEE Wireless Communications, 9*(5), 2002, 76-83.

Performance Characteristics of Cascaded Delay-Line Node Architectures in Single-Path Interconnection Networks

B. Gazi[†] and Z. Ghassemlooy[‡]

[†]Electronics Research Group, School of Engineering, Sheffield Hallam University, Sheffield S1 1WB, UK. Phone: +44 114 2253301, Fax: +44 114 2253433.

[‡]Optical Communications Research Group, School of Engineering and Technology, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK. Phone: +44 191 2274902, Fax: +44 191 2273684.

[*] Correspondence to: Optical Communications Research Group, School of Engineering and Technology, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK.
[†] E-mail: boran.gazi@student.shu.ac.uk
[‡] E-mail: fary.ghassemlooy@unn.ac.uk

SUMMARY

Internally buffered multistage interconnection network (MIN) architectures have been widely used in parallel computer systems and large switching fabrics. Migration from electrical domain to optical domain has raised the necessity of developing node architectures with optical buffers. Cascaded fibber delay line architectures can be seen as possible realizations of output and shared buffering in a 2x2-switching element. These approaches can be used as buffered node architecture in a Banyan like interconnect.

In this paper we investigate and compare these approaches by using simulation methods. Different performance metrics, such as normalized throughput, average packet delay, packet loss rate and buffer utilization have been used under uniform and non-uniform traffic models. Results show that the TC-chain node Banyan network offer an improved normalized throughput and average packet delay performances under both traffic models without disrupting First-In-First-Out order of arrivals. The switched delay-line (SDL) requires fewer switching elements than TC and TTC architectures but at the cost of high packet delay.

KEYWORDS: Cascaded delay line node architectures; interconnection networks

1. INTRODUCTION

Emerging photonic packet switching technologies in communication networks have lead to reconsideration of the existing developments that exploit electrical approaches. Multistage interconnection networks (MINs) that employ optical technologies have been found to be suitable for establishing *N*x*N* ultra-fast packet switches/routers by using 2x2 directional couplers as basic switching elements (SE) [1].

MIN architecture interconnects *N* inputs to *N* outputs through multiple stages of switching elements. Two main types of MIN architectures exist [1]: Internally non-blocking and internally blocking. In the former there are more than one path that connect inputs to the output ports. Thus, packet losses (due to contention in the switching elements) can be prevented by using diversity feature of the interconnect architecture. In internally blocking case, two or more packets from different inputs cannot always be transferred to the outputs of the MIN simultaneously, as resource sharing is required at some stage (in SEs) of the interconnection network. Therefore, only one of the packets is allowed to use the SE output port and the rest of the packets are either dropped or stored in SE buffers. Thus, buffering in the individual network nodes has a crucial impact on the overall performance of the packet switching interconnection network. Due to immaturity of the optical technology, the problem of contention resolution has become even more challenging. However, the fundamental principles applied in electrical domain form the basis for further development in optical communications systems.

Packet switches that employ buffering can be classified into four different groups [2]: Input, output, shared, and recirculation buffering. Input buffering suffers from head of line blocking [3] and it has never been proposed for optical domain.

Shared buffering achieves low packet loss rate by dynamic adjustment of the common buffer space [4] [5]. Exact realisation of shared buffering in optical domain is almost impossible due to immature access feature of optical delay lines (ODLs). Nevertheless similar performances can be readily achieved by sharing the common buffer space in ODLs among output ports in a packet switch. Recirculation buffering has been found practical for both electronic and optical domains [2] [3]. The studies have shown that output buffering achieves the optimal throughput-delay performance [3], which makes it attractive for all-optical implementation.

Possible realisations of output and/or shared buffering are 2x2 multistage feed forward delays (Figure 1): Cascaded Optical Delay-Line (COD) [6], Switched Delay Line [7] [8] [9] and Logarithmic Delay-Line Switch [10]. The common feature of these architectures is that all consist of n stages of delay line modules (d) in cascade. Besides, these architectures can be used as the building block of MIN in order to construct all-optical NxN buffered packet switches.

The purpose of this paper is to illustrate the MIN performance characteristics of 2x2 cascaded delay line switch architectures by means of simulation methods. The next section describes the 2x2 buffered node architectures and compares these architectures based on a stand-alone numerical analysis. Section 3 presents overall system characteristics of the internally buffered optical MIN architecture. Simulation method and the parameters evaluated are described in section 4. Simulation results and the conclusions drawn based on these results are presented in Section 5. Finally, section 6 summaries internally buffered all-optical MIN architecture and finalises the discussions.

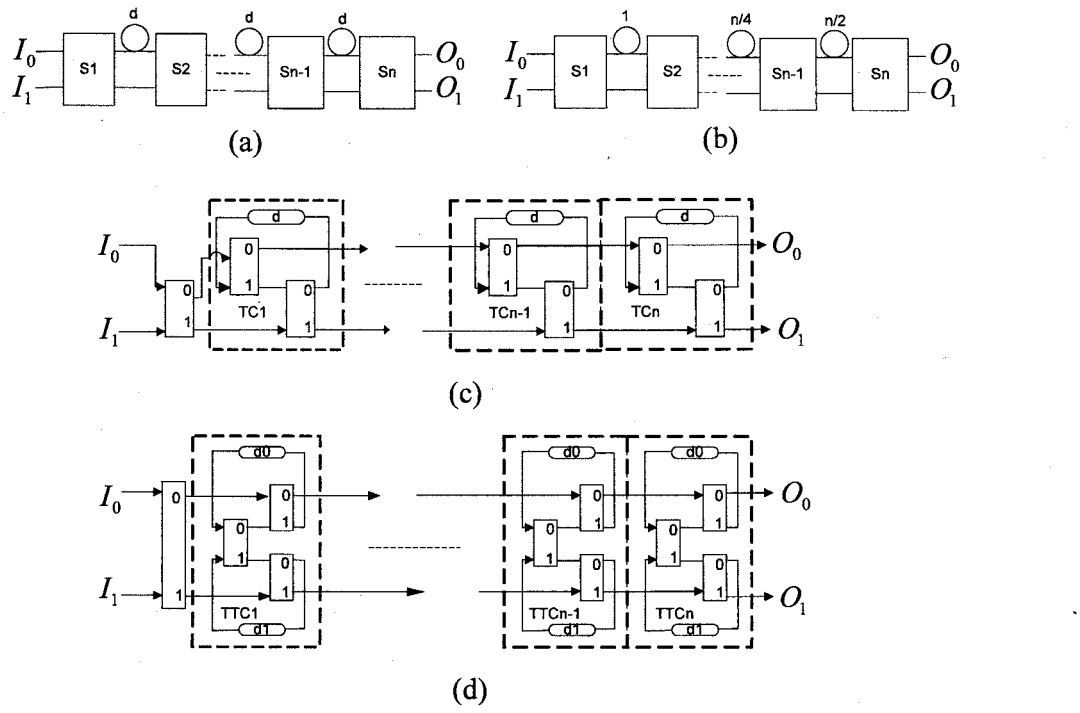


Figure 1 – Cascaded 2x2 switches; (a) SDL, (b) logarithmic delay-line switch, (c) COD track changer, and (d) COD twin track changer

2. CASCADED DELAY LINE NODE ARCHITECTURES

2.1. COD: Track Changer (TC) and Twin Track Changer (TTC)

TC and TTC architectures, proposed by Cruz and Tsai, use the analogy of track changing trains [6]. The purpose of using the track changer is to make packets to move on their original tracks before exiting the module. TC and TTC modules employ integer multiple fibre loop delays, which delay packets for one or more packet length durations. For both architectures, if two packets try to use the same track, one of them is delayed and the other one is allowed to exit from the module output port. In case of TC, if three packets of same type (two input packets and one delayed packet) try to use the same track, then one of them is allowed to exit from the correct output port, one packet is delayed and the last one is deflected to the wrong output port. In case of

TTC, the third packet can be delayed in the second delay line (lower delay line). Similarly, in the TTC architecture, packet deflection can only take place when four packets of same type (two inputs and two packets in two delay lines) try to use the same track. Deflected packets are assumed lost and/or destroyed at the output ports of the switch. For this reason, there is a need to cascade these modules in order to minimize packet losses due to deflection, called *baseline architecture* (Figures 1c and 1d).

The control mechanism of COD architectures is based on a simple self-routing smart switch. A smart switch is a 2x2 basic switch, which employs cross priority when two packets of same type arrive at the inputs of the smart switch. TC baseline architecture has the First-In-First-Out (FIFO) property when the delay line lengths at each TC module are of one packet length. FIFO property for the baseline architecture ensures that when buffer overflow occurs (e.g. two packets arrive at the input ports, the buffer space is fully occupied and all packets in the system are destined to the same output port), it is the oldest packet in the baseline architecture that is lost and the second oldest packet leaves the switch from the correct output port. However, in [6] it was shown that this FIFO property does not hold for the TTC baseline architecture even if the delay line lengths at each module are equal and of one packet length ($d_0 = d_1 = 1$). This is because, in case of buffer overflow, the oldest packet is allowed to exit the baseline architecture and whereas the second oldest packet is lost. In turn order of arrivals is disrupted. It was also shown that $2n$-stage TC in cascade is required to achieve the same packet loss performance as n-stage TTC under uniform network traffic and one packet length delay line lengths.

In [6] it has been shown that by modifying the baseline architecture (i.e. changing delay line lengths) the packet loss rates can be minimized, which greatly

reduces the required number of 2x2 basic smart switches. However, the COD architecture becomes non-work-conserving or idling (i.e. the output ports are poorly utilised) and the FIFO order of packet arrivals is lost.

2.2. Switched Delay-Line (SDL) Switch and Logarithmic Delay-Line Switch

SDL is a 2x2 buffered switch constructed from 2x2 optical switches and fibre delay lines (Figure 1a). The SDL was first proposed by Chlamatac and Fumagalli as a contention resolution unit for node architecture for optical data double ring network [7]. It was also proposed and tested as node architecture for the Manhattan Networks [11]. Control mechanism of SDL is simple and modular. At each stage the control mechanism tries to deliver two different types of packets to the next stage. If the contentions persist until the last stage of SDL switch, the packet that leaves from the wrong output of the switch is assumed as lost packet (deflected). The SDL switch preserves FIFO order if the delay line lengths at each stage are of one packet length ($d_1 = d_2 = \ldots = d_{n-1} = 1$).

The packet loss rate performance of the SDL structure can be improved by proper adjustment of the delay line lengths at each stage. Logarithmic delay-line switch was studied in [10] as an alternative to the SDL. Logarithmic delay-line switch consists of a chain of $\log_2 m + 1$ switches and $\log_2 m$ delay lines, where m is the total buffer capacity of logarithmic delay-line switch. The control mechanism of logarithmic delay-line switch is the same as the SDL. As one can expect due to unequal delay line lengths at each stage, the logarithmic delay line switch suffers from high packet delays, besides FIFO order of packets is lost and the outputs are idle when the input load is very small. However, logarithmic delay-line switch requires less switching elements to achieve the same packet loss rate as SDL [2]. Simply, n-stage

SDL is required to achieve the same loss performance as $\log_2 n$ stages of logarithmic delay-line switch, where n is the total buffer size.

2.3. Stand-alone Comparison

Based on the analytical models developed for the COD [6], SDL and logarithmic delay-line switch [10], an initial stand-alone comparison is carried out to justify the packet loss rate performance of each cascaded fibre delay line switch architecture.

Figure 2a shows the packet loss rate against the traffic load for four optical switches. For this analysis the number of switch stages are selected as four ($n = 4$) for all of the architectures (see Figure 1). That is, four-packet capacity for TC and SDL ($m = n$, where m is the buffer size), and eight-packet storage capacity for the TTC ($m = 2n$) and logarithmic delay-line switch ($m = 2^{n-1}$). The logarithmic delay-line switch performs worse than the TTC due to the idling problem when both have the same amount of buffer space. A switch is non-idling if its output ports are always busy, also called work conserving. It is possible to observe similar relation between the TC and SDL. In Figure 2b, it is shown that for the number of switch stages greater than five ($n > 5$) the logarithmic delay-line switch outperforms other architectures, as fewer 2x2 basic switches are required in achieving similar packet loss rates as other architectures. For instance, TTC requires nineteen (three 2x2 basic switches per stage) 2x2 basic switches to achieve almost the same packet loss performance as the logarithmic delay-line switch with six (one 2x2 basic switch per stage) 2x2 basic switches. Regardless of the low component cost, the logarithmic delay-line switch suffers from high packet delays and output idling due to lengthy delay lines. On the

other hand, the SDL must have at least one packet waiting in ODLs and another packet destined to different output entering the switch in order to be work-conserving.

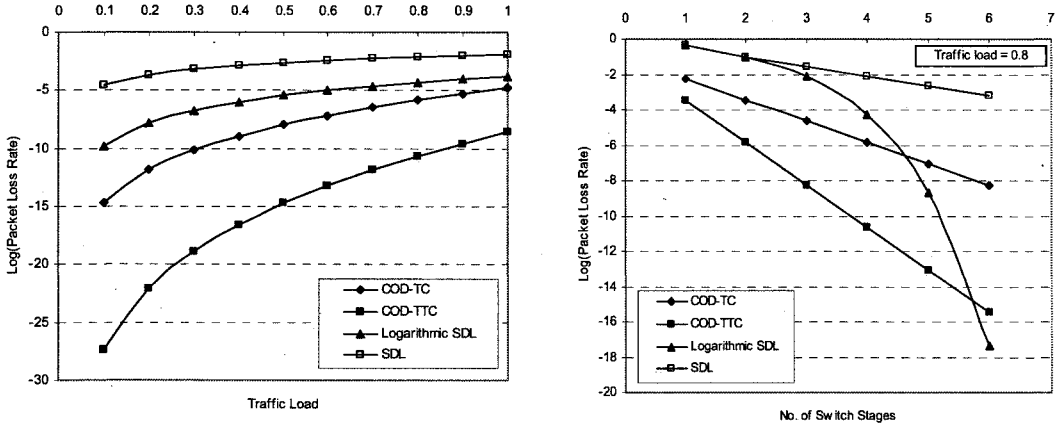


Figure 2 – Packet loss rate versus (a) traffic load and (b) the number of switch stages for COD-TC, COD-TTC, logarithmic SDL, and SDL.

3. CASCADED DELAY LINE NODE BANYAN TOPOLOGY

*N*x*N* cascaded delay-line based Banyan topology connects N inputs to N outputs with $n = \log_2 N$ stages of $N/2$ cascaded delay-line architectures (TC, TTC and SDL; see Figure 3). The optical MIN uses binary self-routing scheme where a packet header contains n bits of address information for selecting which output port to use at each network stage while travelling from the upper stream to the lower stream.

The network operates synchronously, i.e. each time-slot contains a packet or no packet at all. Packet size is fixed with the following header information: n-bit address and packet age (time-slot duration spent in delay lines). The control mechanism of the optical MIN is distributed. There are no buffer sharing mechanisms such as *backpressure*, *restricted backpressure*, *pushout* (see [12] [13]) employed by the network, as packets cannot be stored in the cascaded delay-line architectures

continuously. The maximum delay a packet encounters in a node is bounded by the total duration of the delay lines (t_d) at each module in a chain of length m. Thus, the total maximum possible delay that a packet experiences in the optical MIN architecture is t_d x m x $\log_2 N$. This holds for TTC as well, as packets are not allowed to enter delay lines more than once in a TTC module [6]. However this is only valid if and only if $t_{d0} = t_{d1}$ for every TTC stage.

Packet arrivals at the inputs of the optical MIN is modelled using iid *Bernoulli* process, where in any given time-slot, the probability of arrival of a packet at an input port is p. In uniform traffic, each arriving packet has uniform probability of $1/N$ of being addressed to one of the output ports of the network (where N is the number of inputs/outputs). In case of non-uniform hotspot traffic, one (or some) of the output ports has a higher probability of receiving packets (H_p) than other output ports ($1\text{-}H_p$). In the same way, the probability of non-hotspot ports receiving packets is $(1\text{-}H_p)/(N\text{-}h)$ where h is the number of hotspot ports. In our simulations we are assuming that h = 1.

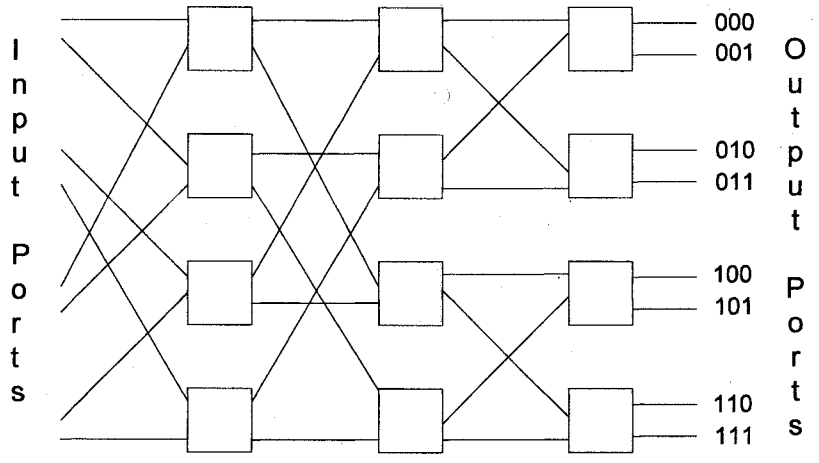


Figure 3 – 8x8 Banyan interconnection network

4. SIMULATION METHOD

Simulations are carried out using a C program. *Bernoulli* random number generator function is used to generate network traffic. Output port address of each packet is generated using C library uniform number generator function. Duration of a single simulation is 50,000 cycles (time-slots). The values computed in first 500 cycles are ignored, as the network needs to reach steady state to achieve realistic results. Simulations are performed using different network sizes, number of switch stages per network node (buffer sizes) and traffic patterns (uniform/non-uniform). Only logarithmic delay-line switch is not considered as the node architecture in our simulations due to its similarity with the SDL switch.

Simulations are carried out to evaluate the normalized network throughput, packet loss rate, the average packet delay and average buffer utilization. The normalized throughput is given as:

$$\tau = \frac{1}{k} \sum_{j=t_0}^{t_1-1} \eta_j \tag{1}$$

Where, η_j is the total number of packets received from all output ports in 1 cycle and k is the total number of cycles within the period t_0-t_1.

Packet loss rate is measured as the ratio of the number of packets lost (deflected) to the total number of packets injected into the network between the instances t_0 and t_1. Average buffer utilization is the total buffer occupancy of each node averaged over t_1-t_0 time-slots. In the same way, the average packet delay in terms of time-slots is computed as follows:

$$\delta = \frac{1}{N \times K} \sum_{j=t_0}^{t_1-1} \sum_{i=0}^{N-1} M_{ij} \tag{2}$$

Where, M_{ij} is the delay of a packet at port i in cycle j. There is a possibility that there may not be a packet in the time-slot at port i during cycle j. Also, lost (deflected) packets are not included when calculating average packet delay.

5. RESULTS AND DISCUSSIONS

TC, TTC and SDL node-based Banyan networks were simulated. Different network sizes, traffic loads and models (uniform and non-uniform hotspot) were considered.

The normalized throughput, average packet delay and packet loss rate performances of the node architectures under uniform network traffic are illustrated in Figure 4. The network and the buffer sizes are 64x64 and 4, respectively. All three architectures display similar throughput performance for network load 0.7. Above 0.7, the CODs display marginally higher throughput. As shown in stand-alone analysis, $2n$-stages of TC node achieve similar packet loss performances as n stages of TTC (Figure 2a). When both of the architectures are used as node architecture in the banyan interconnect, it is possible to observe that both architectures show similar network performances (see Figures 4-8). Hence, it is possible to say that stand-alone behaviour of both TC and TTC architectures still applies when they are integrated into a network. On the other hand SDL display higher packet delays compared with COD-TC and COD-TTC due to the output idling problem (see Figure 4b). Thus resulting in buffer congestion and consequently newly arrived packets are dropped. For this reason, the packet loss rate of the SDL is higher than COD-TC and COD-TTC.

In Figure 5, the effect of node buffer size (average node buffer size of the network) on the normalized throughput, packet loss rate, and the average packet delay under uniform network traffic is depicted for 64x64 Banyan network. In case of all

three-node architectures, for the node buffer size in the range of 0-20, the throughput increases and packet loss rate decreases sharply saturating at node buffer size greater than 140 (Figures 5a and 5b). Hence increasing the buffer size accommodates more space for packets in contention. Nevertheless, employing large buffer size results in an increased packet delays for all three architectures with the SDL showing considerably higher packet delay (Figure 5c). In SDL, linear relation between the buffer depth and the average packet delay is because of the output idling problem. SDL provides full output utilization only when the buffer space becomes fully occupied. It is possible to call this the *pipelining effect.*

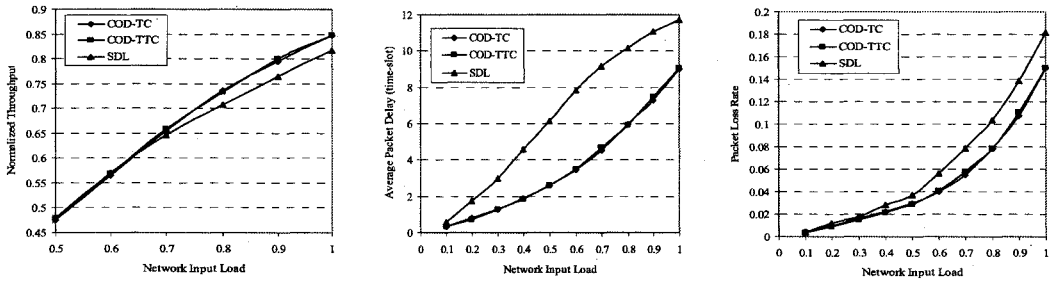


Figure 4 – (a) Normalized throughput, (b) average packet delay (time-slot), and (c) packet loss rate versus network input load (buffer size = 4; network size = 64x64) for COD-TC, COD-TTC and SDL switches

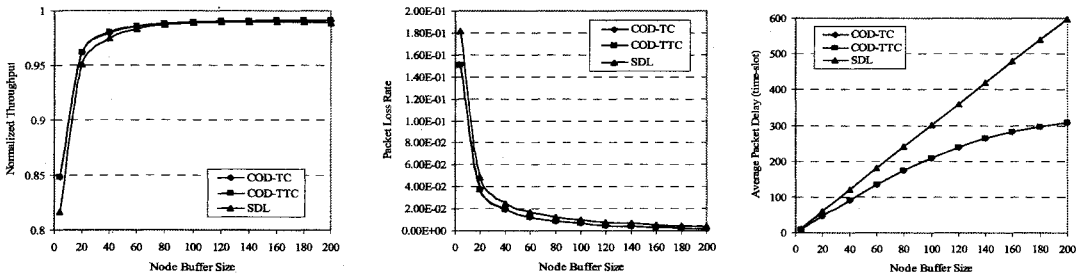


Figure 5 – (a) Normalized throughput, (b) packet loss rate, and (c) average packet delay (time-slot) (traffic load = 1.0; network size = 64x64) for COD-TC, COD-TTC and SDL switches

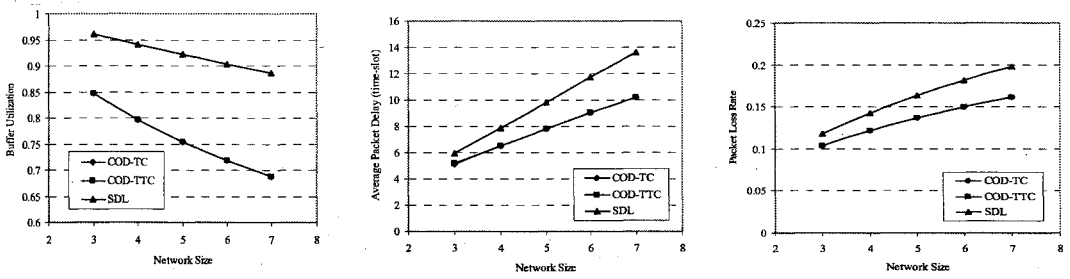


Figure 6 – (a) Buffer utilization (b) average packet delay, and (c) packet loss rate versus network size (traffic load = 1.0; buffer size = 4) for COD-TC, COD-TTC and SDL switches

Under uniform network traffic, as the network size increases (i.e. number of network stages from 3 to 7), the SDL architecture displays higher buffer utilization than COD architectures for the same amount of buffer space of 4 and traffic load of 1 (see Figure 6a). On the other hand the average packet delay and the packet loss rate increase linearly with the network size, with the SDL showing higher profile than CODs (see Figures 6b and 6c).

The average packet delay, buffer utilization, normalized throughput and packet loss rate versus hotspot probability for non-uniform hotspot traffic with buffer size of 4 ODLs (at each node), traffic load of 1.0 and the network size of 64x64 are shown in Figures 7 and 8. From moderate to high hotspot probabilities (0.7 to 1.0) all node architectures suffer highly from the tree-saturation effect. Due to the architectural aspects, it might not be possible to adapt any of the network control mechanisms (such as push-out, backpressure) in order to regulate the effective buffer sharing throughout the entire interconnection network. SDL cannot smooth the traffic to the next stages of the network, because of the problem of internal congestion. For this reason, from light to moderate hotspot probabilities (0.1 to 0.7), it is possible to observe high packet delays (see Figure 7a). Once the network saturates (i.e. high hotspot probability), all three architectures achieve low buffer utilization (see Figure

7b) and similar packet loss rate and normalized throughput performances (see Figure 8). For instance, when the hotspot probability H_p = 1.0, at most one packet leaves the network in every time-slot (see Figure 8a). Thus, the packet loss ratio increases with the rate of hotspot packets and it is not possible to accommodate every hotspot packet in the nodes feeding the hotspot port (see Figure 8b).

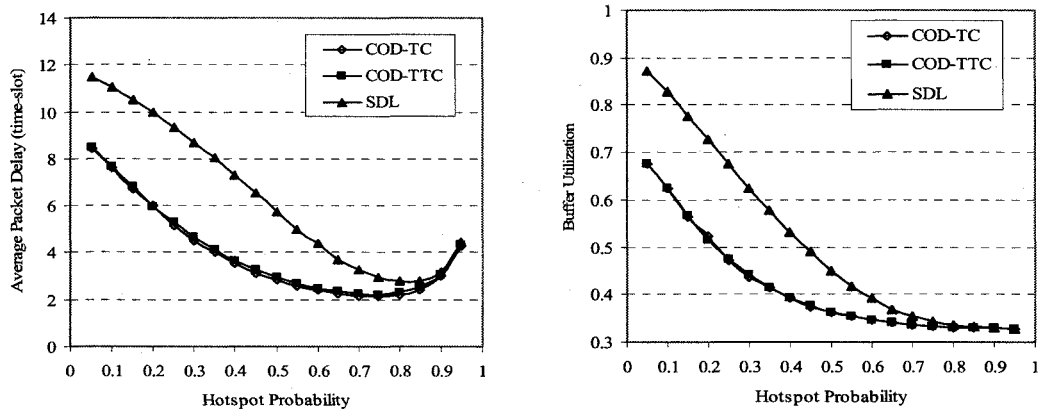


Figure 7 – (a) Average packet delay and (b) buffer utilization versus hotspot probability (traffic load = 1.0; network size = 64x64) for COD-TC, COD-TTC and SDL switches

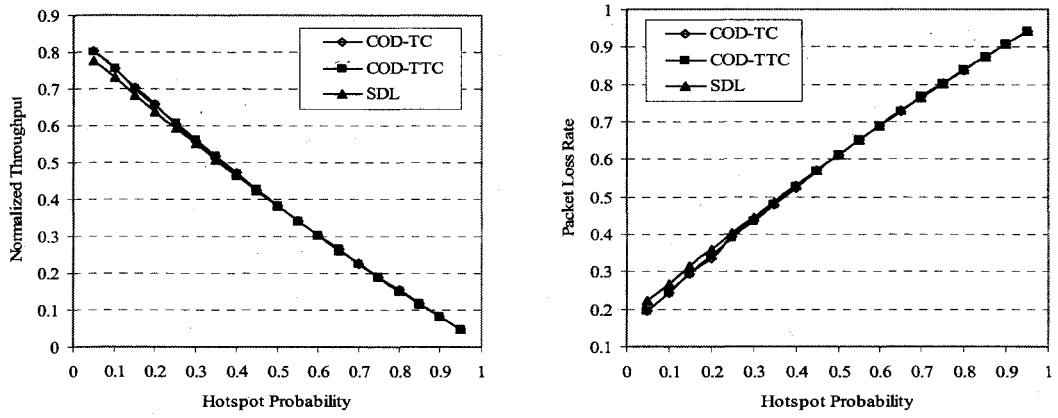


Figure 8 – (a) Normalized throughput, and (b) packet loss rate versus hotspot probability (traffic load = 1.0; network size = 64x64) for COD-TC, COD-TTC and SDL switches

The most common problem of these buffering approaches is that increasing ODL lengths at each switch stage does not necessarily mean increase in the overall performance. However, it is possible to observe better performances in terms of packet loss rate if the ODL lengths at each switch stage are adjusted appropriately (i.e. optimal delay lengths) [6] with the cost of packet order disruption.

6. CONCLUSIONS

A number of numerical analysis and simulations were carried out to investigate the performance and behaviour of cascaded delay-line architectures in an optical MIN with different buffer depths and network sizes under both uniform and non-uniform hotspot network traffic.

The simulation results for the normalized throughput, average packet delay and packet loss rate have shown that SDL node banyan network possesses pipeline feature, where all the stages of the switch have to be filled with packets in order to be work-conserving. In [2] it was suggested that SDL performance could be improved by disrupting the FIFO order. TTC node banyan network can achieve similar performance as TC node banyan with fewer switching components. However, FIFO order of packets cannot be preserved with TTC. On the other hand among TC, TTC and SDL switch architectures, SDL requires the least number of switching components. Both TC and TTC architectures mimic shared buffering in terms of control mechanism and performance metrics.

TC cascaded delay-line node architecture can be used as a node in banyan like networks in order to achieve complete realization of shared buffering if component cost is not important. Instead TTC node architecture can be used to reduce component cost with the cost of FIFO order disruption of packets. The SDL node architecture can

be a better candidate in terms of network scalability and component cost (if and only if the average packet delay is not the main concern). As shown in stand-alone analysis, the logarithmic delay-line switch requires even less components to achieve similar performances as other cascaded delay line switch architectures. However, both SDL and logarithmic delay-line architectures might not be suitable for systems where high delay is not feasible.

For all-optical buffered switch architectures dynamic behaviour is still the main concern. Besides, hierarchical buffer sharing among network nodes for large optical switch interconnects cannot be employed due to (i) feed-forward organization of optical delay lines in packet switching nodes and (ii) inability to continuously recalculate packets within the ODLs. Thus the need for a new control strategies for hierarchical buffer management in all-optical interconnects in order to support asymmetric network traffic and alternative buffered optical switch architectures to cope with multicast traffic, which is the subject of further work.

REFERENCES

1. Okayama H, Okabe Y, Kamijoh T, Sakamoto N. Optical switch array using banyan network. *IEICE Transactions on Communications* 1999; E82-B(2): 365-372.
2. Hunter DK, Chia MC, Andonovic I. Buffering in optical packet switches. *Journal of. Lightwave Technology* 1998; 16: 2081–2094.
3. Hluchyj MG, Karol MJ. Queueing in high performance packet switching. *IEEE Journal on Selected Areas in Communications* 1988; 6(9): 1587-1597.
4. Irland MI. Buffer management in a packet switch. *IEEE Transactions on Communications* 1978; 26(3): 328-337.
5. Arpaci M, Copeland JA. Buffer management for shared-memory ATM switches. *IEEE Communications Surveys & Tutorials* 2000; 3(1): 2-10.
6. Cruz L, Tsai JT. COD: alternative architectures for high speed packet switching. *IEEE/ACM Transactions on Networking* 1996; 4(1): 11-21.
7. Chlamtac I, Fumagalli A. An optical data double ring network. *Journal of High Speed Networks* 1993; 2(4): 355-371.
8. Chlamtac I et al. CORD: contention resolution by delay lines. *IEEE Journal on Selected Areas in Communications* 1996; 14(5): 1014-1029.

9. Chlamtac I, Fumagalli A, Suh C-J. Multibuffer delay line architectures for efficient contention resolution in optical switching nodes. *IEEE Transactions on Communications* 2000; 48(12): 2089-2098.

10. Hunter DK, Cotter D, Ahmad RB, Cornwell WD, Gilfedder TH, Legg PJ, Andonovic I. 2x2 Buffered switch fabrics for traffic routing, merging and shaping in photonic cell networks. *IEEE Journal of Lightwave Technology* 1997; 15(1): 86-101.

11. Chlamtac I, Fumagalli A. An optical switch architecture for manhattan networks. *Journal on Selected Areas in Communications* 1993; 11(4): 550-559.

12. Basak D, Choudhury AK, Hahne EL. Sharing memory in banyan-based ATM switches. *IEEE Journal on Selected Areas in Communications* 1997; 15(5): 881-891.

13. Choudhury AK, Hahne EL. A new buffer management scheme for hierarchical shared memory switches. *IEEE/ACM Transactions on Networking* 1997; 5(5): 728-738.

GLOSSARY OF ABBREVIATIONS

ANN	Artificial Neural Network
ARIMA	Auto-Regressive Integrated Moving Average
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
AWG	Arrayed Wave Guide
BER	Bit Error Rate
CCF	Critical Cell First
CIOQ	Combined Input Output Queueing
CLR	Cell Loss Rate
COD	Cascaded Optical Delay-line
CS	Complete Sharing
CP	Complete Partitioning
CPN	Cognitive Packet Network
CQ	Custom Queueing
DFT	Decay Function Threshold
DoD	Drop on Demand
DQT	Dynamic Queue Thresholds
DT	Dynamic Thresholds
FDM	Frequency Division Multiplexing
FIFO	First In First Out
FL	Fuzzy Logic
FTP	File Transmission Protocol

GAs	Genetic Algorithms
HOL	Head of Line
IPP	Interrupted Poisson Process
ISDN	Integrated Services Digital Network
LQF	Longest Queue First
LRU	Least Recently Used
M/D/1	Poisson arrival, deterministic service time, single server
M/G/1	Poisson arrival, generic service time, single server
M/M/1	Poisson arrival, exponential service time, single server
MIN	Multistage Interconnection Network
MMBP	Markov Modulated Bernoulli Process
MMFM	Markov Modulated Fluid Model
MMPP	Markov Modulated Poisson Process
MPI	Message Passing Interface
MSM	Maximum Size Matching
MSN	Manhattan Street Networks
MWM	Maximum Weight Matching
OCF	Oldest Cell First
ODL	Optical Delay Line
OSU	Optical Storage Unit
OTDM	Optical Time Division Multiplexing
PIM	Parallel Iterative Matching
POT	Push Out with Thresholds
PQ	Priority Queueing
PSPP	Partial Sharing Partial Partitioning

PVM	Parallel Virtual Machine
PVT	Push out with Virtual Thresholds
QoS	Quality of Service
RAM	Random Access Memory
RL	Reinforcement Learning
RPA	Reservation with Preemption and Acknowledgment
RSVP	Resource Reservation Protocol
SBS	Shared Buffer Switch
SDM	Space Division Multiplexing
SLA	Stochastic Learning Automata
SMA	Sharing with Minimum Allocation
SMQMA	Sharing with Maximum Queue Length and Minimum Allocation
SMXQ	Sharing with Maximum Queue Lengths
SOA	Semiconductor Optical Amplifier
SOC	Switch-on-Chip
TC	Track Changer
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TID	Task Identification
TSD	Threshold based Selective Drop
TTC	Twin Track Changer
TWC	Tunable Wavelength Converters
VBR	Variable Bit Rate
VoIP	Voice over IP
VOQ	Virtual Output Queueing

WDM	Wavelength Division Multiplexing
WFQ	Weighted Fair Queueing
WRR	Weighted Round Robin

BIBLIOGRAPHY

Adas, A. (1997) 'Traffic models in broadband networks', *IEEE Communications Magazine*, 35(7), 82-89.

Addie, R., Zukerman, M. and Neame, T. (1998) 'Broadband traffic modeling: Simple solutions to hard problems', *IEEE Communications Magazine*, 36(8), 88-95.

Als, A. and Ghassemlooy, Z. (2004) 'Mathematical modeling of an all-optical buffer suitable for OTDM transmission systems', *Optics Communications*, 238(4-6), 281-290.

Arpaci, M. and Copeland, J.A. (2000) 'Buffer management for shared-memory ATM switches', *IEEE Communications Surveys and Tutorials*, 3(1), 2-10.

Ascia, G., Catania, V. and Panno, D. (2002) 'An efficient buffer management policy based on an integrated Fuzzy-GA approach', *Proceedings of IEEE INFOCOM*, New York, 23-27 June, 2002, 1042-1048.

Ascia, G., Catania, V. and Panno, D. (2005) 'An evolutionary management scheme in high-performance packet switches', *IEEE/ACM Transactions on Networking*, 13(2), 262-275.

Atiquzzaman, M. and Akhtar, M.S. (1995) 'Performance of buffered multistage interconnection networks in a non uniform traffic environment', *Journal of Parallel and Distributed Computing*, 30(1), 52-63.

Basak, D., Choudhury, A.K. and Hahne, E.L. (1997) 'Sharing memory in banyan-based ATM switches', *IEEE Journal on Selected Areas in Communications*, 15(5), 881-891.

Barri, P. and Goubert, J.A.O. (1991) 'Implementation of a 16 to 16 switching element for ATM exchanges', *IEEE Journal on Selected Areas in Communications*, 9(5), 751-757.

Bhatt, S., Fujimoto, R.M., Ogielski, A. and Perumalla, K. (1998) 'Parallel simulation techniques for large-scale networks', *IEEE Communications Magazine*, 36(8), 42-47.

Boyan, J.A. and Littman, M.L. (1994) 'Packet routing in dynamically changing networks: a reinforcement learning approach', *Advances in Neural Information Processing Systems* 6, Morgan Kaufmann Publishers, Inc., 671–678.

Cherry, S. (2004) 'Edholm's law of bandwidth', *IEEE Spectrum,* July, 58-60.

Chlamtac, I. and Fumagalli, A. (1993a) 'An optical switch architecture for Manhattan networks', *IEEE Journal on Selected Areas in Communications*, 11(4), 550-559.

Chlamtac, I. and Fumagalli, A. (1993b) 'An optical data double ring network', *Journal of High Speed Networks*, 2(4), 355-371.

Chlamtac, I. and Fumagalli, A. (1994) 'Quadro-star: a high performance optical WDM star network', *IEEE Transactions on Communications*, 42(8), 2582-2591.

Chlamtac, I., Fumagalli, A., Kazovsky, L.G., Melman, P., Nelson, W.H., Poggiolini, P., Cerisola, M., Choudhury, A.N.M., Fong, T.K., Hofmeister, R.T., Lu, C. L., Mekkittikul, A., Sabido IX, D.J.M., Suh, C. J. and Wong, E.W.M. (1996) 'Cord: contention resolution by delay lines', *IEEE Journal on Selected Areas in Communications*, 14(5), 1014-1029.

Chlamtac, I., Fumagalli, A. and Chang-Jin, S. (2000) 'Multibuffer delay line architectures for efficient contention resolution in optical switching nodes', *IEEE Transactions on Communications*, 48(12), 2089-2098.

Chou, L.-D. and Wu, J.-L.C. (1995) 'Buffer management using Genetic Algorithms and Neural Networks', *Proceedings of the IEEE GLOBECOM*, USA, 7-10 November, 1333-1337.

Choudhury, A.K. and Hahne, E.L. (1997) 'A new buffer management scheme for hierarchical shared memory switches', *IEEE/ACM Transactions on Networking*, 5(5), 728-738.

Choudhury, A.K. and Hahne, E.L. (1998) 'Dynamic queue length thresholds for shared-memory packet switches', *IEEE/ACM Transactions on Networking*, 6(2), 130-140.

Chu, Y.S., Yang, R.B., Wu, C.S. and Liang, M.C. (2002) 'Partial sharing and partial partitioning buffer management scheme for shared buffer packet switches', *IEICE Transactions on Communications*, E85-B(1), 79-88.

Chuang, S.T., Goel, A., McKeown, N. and Prabhakar, B. (1999) 'Matching output queueing with a combined input/output-queued switch', *IEEE Journal on Selected Areas in Communications*, 17(6), 1030-1039.

Chuang, S.T., Iyer, S. and McKeown, N. (2005) 'Practical algorithms for performance guarantees in buffered crossbars', *Proceedings of IEEE INFOCOM*, Florida, 13-17 March, 2005, 981-991.

Cidon, I., Georgiadis, L., Guerin, R. and Khamisy, A. (1995) 'Optimal buffer sharing', *IEEE Journal on Selected Areas in Communications*, 13(7), 1229-1240.

Cisco Systems (2000) *Internetworking technologies handbook*, 3rd edition, CISCO Press.

Cruz, R.L. and Tsai, J.T. (1996) 'COD: alternative architectures for high speed packet switching', *IEEE/ACM Transactions on Networking*, 4(1), 11-21.

Deng, K.L., Runser, R.J., Toliver, P., Glesk, I. and Prucnal, P.R. (2000) 'A Highly-scalable, rapidly-reconfigurable, multicasting-capable, 100-Gb/s photonic switched interconnect based upon OTDM technology', *Journal of Lightwave Technology*, 18(12), 1892-1904.

Dorigo, M., Maniezzo, V. and Colorni, A., (1996) 'The ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1), 29-41.

Engbersen, A.P.J. (2003) 'Prizma switch technology', *IBM Journal of Research and Development*, 47(2/3), 195-209.

Fan, R., Ishii, A., Mark, B., Ramamurthy, G. and Ren, Q. (1999) 'An optimal buffer management scheme with dynamic thresholds', *Proceedings of IEEE GLOBECOM*, Rio de Janeiro, 5-9 December, 1999, 631-637.

Ferra, H.L., Lau, K., Leckie, C. and Tang, A. (2003) 'Applying reinforcement learning to packet scheduling in routers', *Proceedings of the Fifteenth Conference on Innovative Applications of Artificial Intelligence*, Acapulco, 12-14 August, 2003, 79-84.

Ferrari, A.J. (1997) 'Jpvm: network parallel computing in Java', *Technical Report, UMI Order Number: CS-97-29*, University of Virginia.

Flower, J. and Kolawa, A. (1994) 'Express is not just a message passing system: current and future directions in Express', Parallel Computing, 20(4), 597-614.

Frost, V.S. and Melamed, B. (1994) 'Traffic modeling for telecommunications networks', *IEEE Communications Magazine*, 32(3), 70-81.

Gabriagues, J.M. and Jacob, J.B. (1995) 'OASIS: a high-speed photonic ATM switch – results and perspectives', *Proceedings of the 15th International Switching Symposium*, Berlin, 23-28 April, 1995, 457-461.

Gambini, P., Renaud, M., Guillemot, C., Callegati, F., Andonovic, I., Bostica, B., Chiaroni, D., Corazza, G., Danielsen, S.L., Gravey, P., Hansen, P.B., Henry, M., Janz, C., Kloch, A., Krahenbuhl, R., Raffaelli, C., Schilling, M., Talneau, A. and Zucchelli, L. (1998) 'Transparent optical packet switching: network architecture and demonstrators in the KEOPS project', *IEEE Journal on Selected Areas in Communications*, 15(7), 1245-1259.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V.S. (1994) *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press.

Gelenbe, E., Lent, R., and Nunez, A. (2004) 'Self-aware networks and QoS', *Proceedings of the IEEE*, 92(9), 1478-1489.

Gelernter, D. (1985) 'Generative communication in Linda', *ACM Transactions on Programming Languages and Systems*, 7(1), 80-112.

Giacopelli, J.N., Hickey, J.J., Marcus, W.S., Sincoslie, W.D. and Littlewood, M. (1991) 'Sunshine: a high performance self-routing broadband packet switch architecture', *IEEE Journal on Selected Areas of Communications*, 9(8), 1289-1298.

Gonzalez-Herraez, M., Song, K.-Y. and Thevenaz, L. (2005) 'Optically controlled slow and fast light in optical fibers using stimulated Brillouin scattering', *Applied Physics Letters*, 87(8), 081113.

Gross, D. and Harris, C.M. (1998) *Fundamentals of Queueing Theory*, 3rd Edition, John Wiley & Sons, Inc.

Guizani, M. (1997) 'ATM architectures using optical technology: an overview of switching, buffering and multiplexing', *International Journal of Network Management*, 7(4), 198-220.

Guizani, M. and Al-Fuqaha, A.I. (2001) 'A new queueing strategy for large scale ATM switches', *IEEE Communications Magazine*, December 2001, 142-146.

Harai, H. and Murata, M. (2006) 'High-speed buffer management for 40 gb/s-based photonic packet switches', *IEEE/ACM Transactions on Networking*, 14(1), 191-204.

Haas, Z. (1992) 'The staggering switch: an almost-all optical packet switch', *OFC Technical Digest Series*, 132.

Hluchyj, M. and Karol, M. (1988) 'Queueing in high-performance packet switching', *IEEE Journal on Selected Areas in Communications*, 6(9), 1587-1597.

Huang, A. and Knauer, S. (1984) 'Starlite: a wideband digital switch', *Proceedings of IEEE GLOBECOM*, Atlanta, December, 1984, 121-125.

Hunter, D.K. and Andonovic, I. (1993) 'Optical contention resolution and buffering module for ATM networks', *IEEE Electronics Letters*, 29(3), 280-281.

Hunter, D.K. (1995) 'ATM switch and crossconnect architectures', *IEE Tutorial Colloquium on Optics and ATM*, London, paper 3, digest 1995/031, 3/1-3/5.

Hunter, D.K., Cotter, D., Ahmad, R.B., Cornwell, W.D., Gilfedder, T.H., Legg, P. J. and Andonovic, I. (1997) '2×2 buffered switch fabrics for traffic routing, merging, and shaping in photonic cell networks', *Journal of Lightwave Technology*, 15(1). 86-101.

Hunter, D.K., Chia, M.C. and Andonovic, I. (1998a) 'Buffering in optical packet switches', *IEEE/OSA Journal of Lightwave Technology*, 16(12), 2081-2094.

Hunter, D.K., Cornwell, W.D., Gilfedder, T.H., Franzen, A. and Andonovic, I. (1998b) 'SLOB: a switch with large optical buffers for packet switching', *IEEE/OSA Journal of Lightwave Technology*, 16(10), 1725-1736.

Hunter, D.K., Nizam, M.H.M., Chia, M.C., Andonovic, I., Guild, K.M., Tzanakaki, A., O'Mahony, M.J., Bainbridge, L.D., Stephens, M.F.C., Penty, R.V. and White, I.H. (1999) 'WASPNET: a wavelength switched packet network', *IEEE Communications Magazine*, 37(3), 120-129.

Iliadis, I. and Denzel, W.E. (1993) 'Analysis of packet switches with input and output queueing', *IEEE Transactions on Communications*, 41(5), 731-740.

Irland, M. (1978) 'Buffer management in a packet switch', *IEEE Transactions on Communications*, COM-26(3), 328-337.

Kamoun, F. and Klienrock, L. (1980) 'Analysis of shared finite storage in a computer network node environment under general traffic conditions', *IEEE Transactions on Communications*, COM-28(7), 992-1003.

Karol, M.J., Hluchyj, M. and Morgan, S.P. (1987) 'Input versus output queueing on a space-division packet switch', *IEEE Transactions on Communications*, COM-35(12), 1347-1356.

Karol, M.J. (1992) 'Queueing in optical packet switches', *Proceedings of SPIE Vol. 1787: Multigigabit Fiber Communications*, Boston, September, 1992, 192-203.

Karol, M.J. (1993) 'Shared-memory optical packet (ATM) switch', *Proceedings of SPIE Vol. 2024: Multigigabit Fiber Communication Systems*, San Diego, 14-16 July, 1993, 212-222.

Karol, M., Golestani, J. and Lee, D. (2003) 'Prevention of deadlocks and livelocks in lossless backpressured packet networks', *IEEE/ACM Transactions on Networking*, 11(6), 923-934.

Kesselman, A. and Mansour, Y. (2002) 'Harmonic buffer management policy for shared memory switches', *Proceedings IEEE INFOCOM*, New York, June, 2002, 1615-1622.

Kozaki, T., Endo, N., Sakurai, Y., Matsubara, O., Mizukami, M. and Asano, K. (1991) '32×32 shared buffer type ATM switch VLSI's for B-ISDN's', *IEEE Journal on Selected Areas of Communications*, 9(8), 1239-1247.

Leonardi, E., Mellia, M., Neri, F. and Marsan, M.A. (2001) 'On the stability of input-queued switches with speed-up', *IEEE/ACM Transactions on Networking*, 9(1), 104-118.

Liew, S.C. (1994) 'Performance of various input-buffered and output-buffered ATM switch design principles under bursty traffic: simulation study', *IEEE Transactions on Communications*, 42(2/3/4), 1371-1379.

Marsan, M.A., Bianco, A., Leonardi, E. and Milia, L. (1999) 'RPA: a flexible scheduling algorithm for input buffered switches', *IEEE Transactions on Communications*, 47(12), 1921-1933.

Marsan, M.A., Bianco, A., Giaccone, P., Leonardi, E. and Neri, F. (2001) 'Input-queued router architectures exploiting cell-based switching fabrics', *Computer Networks*, 37(5), 541-559.

McKeown, N.W. (1995) *Scheduling algorithms for input-queued cell switches*, thesis (Ph.D.), University of California at Berkeley.

McKeown, N., Anantharam, V. and Walrand, J. (1996) 'Achieving 100% throughput in an input-queued switch', *IEEE Transactions on Communications*, 47(8), 1260-1267.

McKeown, N. and Anderson, T.E. (1998) 'A quantitative comparison of iterative scheduling algorithms for input-queued switches', *Computer Networks*, 30(24), 2309-2326.

McKeown, N. (1999) 'The iSLIP scheduling algorithm for input-queued switches', *IEEE/ACM Transactions on Networking*, 7(2), 188-201.

Minkenberg, C. and Engbersen, T. (2000) 'A combined input and output queued packet-switched system based on PRIZMA switch-on-a-chip technology', *IEEE Communications Magazine*, 38(12), 70-77.

Mukherjee, B. (2000) 'WDM optical communication networks: progress and challenges', *IEEE Journal on Selected Areas in Communications*, 18(10), 1810-1824.

Mun, Y. and Youn, H.Y. (1992) 'Performance analysis of finite buffered multistage interconnection networks', *Proceedings of the ACM/IEEE Conference on Supercomputing*, Minneapolis, 16-20 November, 1992, 718-727.

Nicol, D.M. and Liu, J. (2002) 'Composite synchronization in parallel discrete-event simulation', *IEEE Transactions on Parallel and Distributed Systems*, 13(5), 433-446.

Nojima, S., Tsutsui, E., Fukuda, H. and Hashimoto, M. (1987) 'Integrated services packet network using bus matrix switch', *IEEE Journal on Selected Areas in Communications*, 5(8), 1284-1292.

Obaidat, M.S., Papadimitriou, G.I., Pomportsis, A. S. and Laskaridis, H. S. (2002) 'Learning automata-based bus arbitration for shared-medium ATM switches', *IEEE Transactions on Systems, Man, and Cybernetics*, 32B(6), 815-820.

Pacheco, P.S. (1996) *Parallel Programming with MPI*, Morgan Kaufmann Publishers Inc.

Saadawi, T.N., Ammar, M.H. and Hakeem, A.E. (1994) *Fundamentals of telecommunication networks*, John Wiley and Sons Inc.

Schaar, M. van der, Krishnamachari, S., Sunghyun, C. and Xiaofeng, X. (2003) 'Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs', *IEEE Journal on Selected Areas in Communications*, 21(10), 1752-1763.

Sharma, S. and Viniotis, Y. (1999) 'Optimal buffer management policies for shared-buffer ATM switches', *IEEE/ACM Transactions on Networking*, 7(4), 575-587.

Stunkel, C.B., Shea, D.G., Abali, B., Atkins, M.G., Bender, C.A., Grice, D.G., Hochschild, P., Joseph, D.J., Nathanson, B.J., Swetz, R.A., Stucke, R.F., Tsao, M. and Varker, P.R. (1995) 'The SP-2 high-performance switch', *IBM Systems Journal*, 34(2), 185-204.

Sutton, R.S. and Barto, A.G. (1998) *Reinforcement learning: an introduction*, MIT Press.

Thareja, A.K. and Tripathi, S.K. (1984) 'Buffer sharing in dynamic load environment', *Proceedings of IEEE INFOCOM*, San Francisco, 9-12 April, 1984, 369-375.

Theimer, T.H., Rathgep, E.P. and Huber, M.N. (1991) 'Performance analysis of buffered banyan networks', *IEEE Transactions on Communications*, 39(2), 269-277.

Thomas, G. and Man, J. (1993) 'Improved windowing rule for input buffered packet switches', *IEE Electronics Letters*, 29(4), 393-395.

Tipper, D. and Sundareshan, M.K. (1988) 'Adaptive policies for optimal buffer management in dynamic load environments', *IEEE Magazine*, March 1988, 535-544.

Walrand, J. and Varaiya, P. (1996) *High-performance communication networks*, Morgan Kaufmann Publishers, Inc.

Wei, S.X., Coyle, E.J. and Hsiao, M.T.T. (1991) 'An optimal buffer management policy for high-performance packet switching', *Proceedings of IEEE GLOBECOM*, Arizona, 2-5 December, 1991, 924-928.

Wu, G.L. and Mark, J.W. (1995) 'A buffer allocation scheme for ATM networks: complete sharing based on virtual partition', *IEEE/ACM Transactions on Networking*, 3(6), 660-670.

Vlasov, Y.A., O'Boyle, M., Hamann, H. and McNab, S.J. (2005) 'Active control of slow light on a chip with photonic crystal waveguides', *Nature Letters*, 438(7064), 65-69.

Yang, J.P., Chu, Y.S. and Liang, M.C. (2003a) 'Threshold-based selective drop for shared buffer packet switches', *IEEE Communications Letters*, 7(4), 183-185.

Yang, R.B., Chu, Y.S., Wu, C.S. and Liang, M.C. (2003b) 'Pushout with virtual thresholds buffer management scheme in a shared buffer ATM switch', *International Journal of Network Management*, 13(2), 147-154.

Yeh, Y.S., Hluchyj, M.G. and Acampora, A.S. (1987) 'The knockout switch: a simple, modular architecture for high-performance packet switching', IEEE *Journal on Selected Areas in Communications*, SAC-5(8), 1274-1283.

Yin, J. and Agrawal, D.P. (2004) 'Optimal packet size in error-prone channel for IEEE 802.11 distributed coordination function', *Proceedings of IEEE Wireless Communications and Networking Conference*, Atlanta, 21-25 March, 2004, 1654-1659.

Yousefi'zadeh, H. (2002) 'A neural-based technique for estimating self-similar traffic average queueing delay', *IEEE Communications Letters*, 6(10), 419- 421.

Zegura, E.W. (1993) 'Architectures for ATM switching systems', *IEEE Communications Magazine*, 31(2), 28-37.

Zhou, B. and Atiquzzaman, M. (2002) 'A performance comparison of four buffering schemes for multistage interconnection networks', *International Journal of Parallel and Distributed Systems and Networks*, 5(1), 17-25.